PERGAMON

Pattern Recognition 35 (2002) 895–914

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Optical character recognition of the Orthodox Hellenic Byzantine Music notation

Velissarios G. Gezerlis , Sergios Theodoridis *

Department of Informatics and Telecommunications, University of Athens, Division of Communication and Signal Processing, Panepistimioupolis, TYPA Buildings, 157 84, Athens, Greece

Received 21 October 1999; accepted 17 April 2001

Abstract

In this paper we present for the first time, the development of a new system for the off-line optical recognition of the characters used in the orthodox Hellenic Byzantine Music notation, that has been established since 1814. We describe the structure of the new system and propose algorithms for the recognition of the 71 distinct character classes, based on Wavelets, 4-projections and other structural and statistical features. Using a nearest neighbor classifier, combined with a post classification schema and a tree-structured classification philosophy, an accuracy of 99.4% was achieved, in a database of about 18,000 Byzantine character patterns that have been developed for the needs of the system.
© 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Optical music recognition; Off-line character recognition; Byzantine Music; Byzantine Music notation; Wavelets; Projections; Neural networks; Contour processing; Nearest neighbor classifier; Byzantine Music database

1. Introduction

Byzantine Music is the type of music that was developed, formed and cultured, in the area of the Hellenic Orthodox Christian Church, flourished mainly in the Byzantine epoch, continued its course in the afterwards metabyzantine years and evolved, throughout the years, to the form which is known today, serving always the warship needs of the Orthodox Church.

Byzantine Music originates from the Ancient Hellenic Music that evolved throughout the centuries into the Classical or Western music, developed mainly in west Europe, and into the Hellenic Byzantine Music of our days, as well as the traditional folk music of Hellas [1,2].

A lot of research effort has been invested over the last 40 years, for the development of systems that would be able to understand and recognize the western music notes. In the mid-1960s, the first published work by Pruslin [3] was carried out at Massachusetts Institute of Technology, concerning an OMR (optical music recognition) system which was the first approach in recognizing musical notes. Prerau [4–6] continued the work of Pruslin, including in the system more symbols and using new techniques. A survey of different styles of music was undertaken and a database built. This database with bitmaps of western music notes was the main heuristic used for the classification of notes.

These were the first attempts in the development of the first OMR system for recognizing the western or classical music notation. From then on, in the interval of about 35 years, many papers have been written, more than 20 projects have taken place and many hardware and software systems have been developed with the ability of music understanding, music recognition and music performance [7,8].

The goal of this work is to develop a new optical recognition system for the notation of the Hellenic Orthodox

* Corresponding author. Tel.: +30-1-727-5328; fax: +30-1-72-19-561.
E-mail addresses: gbelis@di.uoa.gr (V.G. Gezerlis), stheodor@di.uoa.gr (S. Theodoridis).

Byzantine Music. We will present an OBMR (optical Byzantine Music recognition) system that constitutes the first effort in the area of the Byzantine Music [9,35].

The Byzantine Music notation, i.e., the way of writing a *psalm* (a piece of Byzantine Music), presents a particular interest, not only for the great variety of the used symbols, but also for the way the symbols are combined, making *semantic musical groups* of different meanings. So, for the development of such a system we must have in mind the morphological structure of the Byzantine Musical script in order to use the appropriate recognition techniques.

The OBMR system is an off-line optical recognition system and consists of three different and independent stages. First of all, we scan the page of a Byzantine Music piece or text, which is called *psalm*, at 300 dpi. The OBMR system consists of (a) the *segmentation* stage, (b) the *recognition* stage (including a database of all symbols), and (c) the *semantic musical group recognition* stage (including a grammar, and a database of the symbols of the Byzantine Music, as groups).

This paper is focused on the recognition stage. Towards this goal, various feature generation techniques were employed, some of which are new and others which have been used for recognizing Arabic and Chinese characters [10–12].

The paper is structured as follows. In the second section the structure and the morph of the Byzantine Music (BM) and the Byzantine Music notation (BMN) is discussed. In Section 3 we describe the structure of the OBMR system as well as the database of the bitmaps of the BMN. In Section 4, details of the recognition stage are given and all the algorithms for the preprocessing and the features developed and used are described. Features based on wavelet transform, 4-projections (vertical, horizontal, left and right diagonal) and other structural and statistical features have been derived from the contour processing. A notable characteristic of the BMN is that many characters are very similar. This makes the recognition task particularly demanding. In the last section we describe the classification schema, which follows a tree-structure philosophy and the classification is based on nearest neighbor classifiers that gives better results compared to other classifiers such as Neural networks and the experimental results of the recognition part are given. To improve the performance, a post classification schema has also been developed that resolves certain symbol confusions.

2. Byzantine Music and Byzantine Music notation

Byzantine Music is a special type of phonetic music, having its own notation, its own way of performance as well as its own usage, serving always, as we mentioned before, the warship needs of the Orthodox Church. An example of the morph and the structure of a *psalm* of the Byzantine Music is given in Fig. 1.

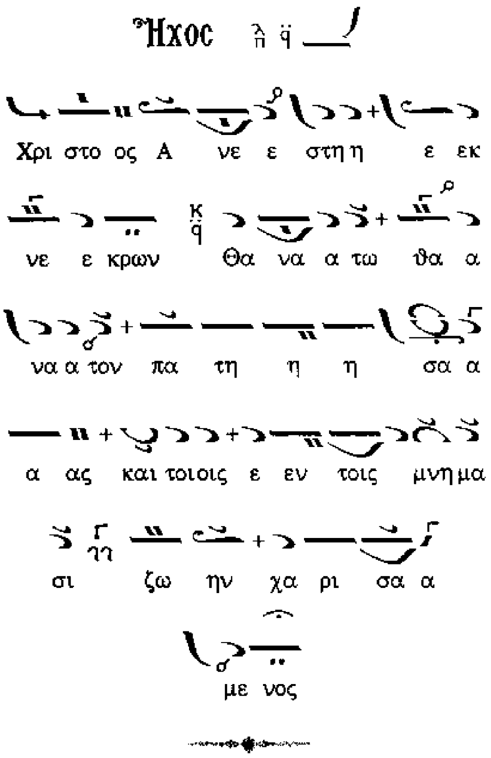
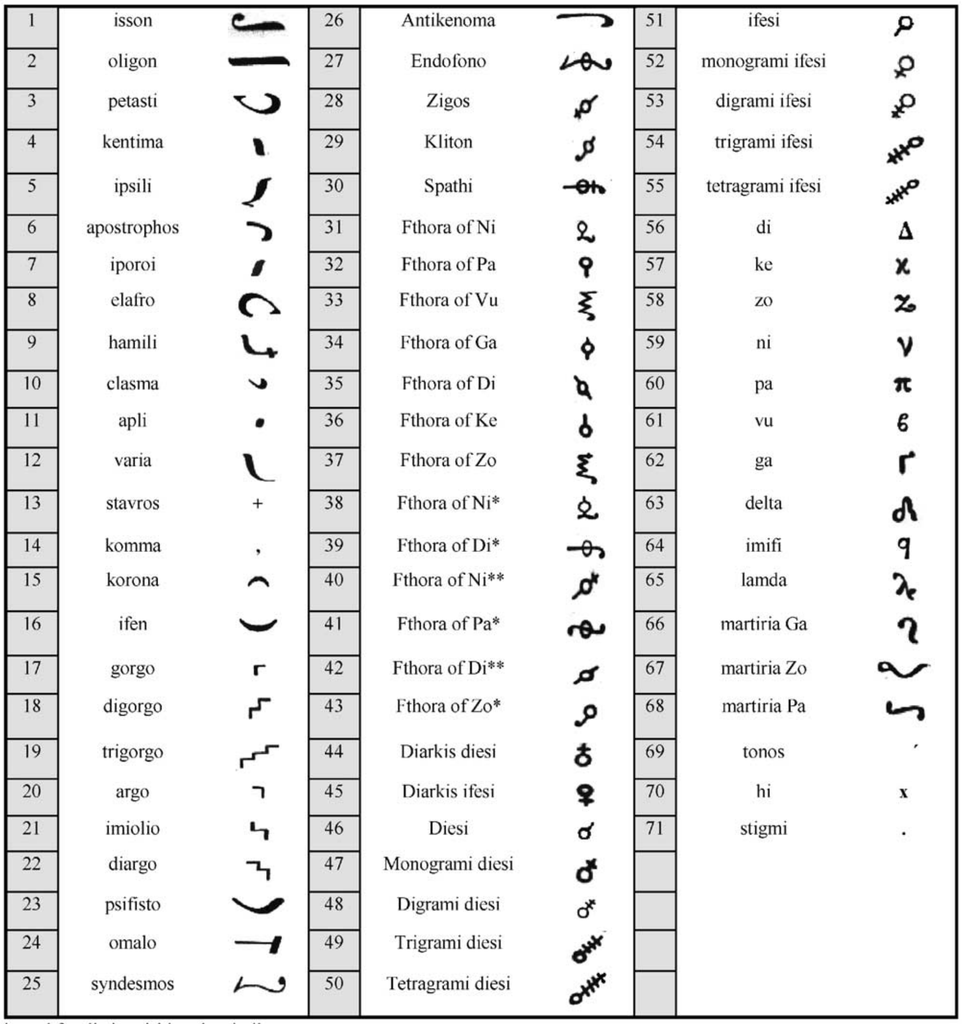

Ἦχος
Χρι στο ος Α νε ε στη η ε εκ
νε ε κρων Θα να α τω θα α
να α τον πα τη η η σα α
α ας και τοιοις ε εν τοις μνη μα
σι ζω ην χα ρι σα α
με νος

Fig. 1. A psalm of the Byzantine Music which is called "Christ is Risen".

As we can notice from Fig. 1, each line of the psalm consists of two parts or one pair. The first one is the Byzantine music Notation, called *chant*. This consists of about 71 distinct symbols (characters), which are combined to form groups of symbols, each one having its own semantic musical meaning and its own musical performance. In the BM more than 2500 groups of such symbols exist. The second part of each line, corresponding to the singing part of the psalm, consists of the spoken words and syllables written in the Hellenic alphabet.

This kind of Notation, which is called "*Analytical script*" of BM, is the newest one completed and established for use into the Orthodox Church in 1814 [1]. In this new analytical script of BM, which is in use in our days, many symbols from the old "*Shortened script*", being in use before 1814, have been retained. Thousands of musical handwritings have been written in this old script of BM that are still unpublished in the libraries of Holy Mountain and other Orthodox Monasteries. On the other hand, many musical books have been published in the new analytical script.

Table 1
The 71 distinct symbols of the Byzantine Music notation



1	isson		26	Antikenoma		51	ifesi	
2	oligon		27	Endofono		52	monogrami ifesi	
3	petasti		28	Zigos		53	digrami ifesi	
4	kentima		29	Kliton		54	trigrami ifesi	
5	ipsili		30	Spathi		55	tetragrami ifesi	
6	apostrophos		31	Fthora of Ni		56	di	
7	iporoi		32	Fthora of Pa		57	ke	
8	elafro		33	Fthora of Vu		58	zo	
9	hamili		34	Fthora of Ga		59	ni	
10	clasma		35	Fthora of Di		60	pa	
11	apli		36	Fthora of Ke		61	vu	
12	varia		37	Fthora of Zo		62	ga	
13	stavros	+	38	Fthora of Ni*		63	delta	
14	komma	,	39	Fthora of Di*		64	imifi	
15	korona		40	Fthora of Ni**		65	lamda	
16	ifen		41	Fthora of Pa*		66	martiria Ga	
17	gorgo		42	Fthora of Di**		67	martiria Zo	
18	digorgo		43	Fthora of Zo*		68	martiria Pa	
19	trigorgo		44	Diarkis diesi		69	tonos	ʹ
20	argo		45	Diarkis ifesi		70	hi	x
21	imiolio		46	Diesi		71	stigmi	.
22	diargo		47	Monogrami diesi				
23	psifisto		48	Digrami diesi				
24	omalo		49	Trigrami diesi				
25	syndesmos		50	Tetragrami diesi				

* used for distinguishing the similar names

2.1. Characteristics of the BMN

- The BMN is being written from left to right, so the optical reading will follow the same direction.
- The characters of the notation are not distinguished in caps or small ones, but they are always as they appear in Table 1.
- An additional feature of the BMN is that the characters do not touch each other in a musical text (see Fig. 1) and are always separated. If two characters of the BMN touch each other, then this is due to an error, during the printing or writing of the melody. In the following discussion, we will consider that characters are well separated.
- As we can observe from Fig. 1, the characters of BMN are combined so that they are in left or right, above or below and left or right diagonal position, and in general to many potential positions.
- Below each line of chant, there are words or letters of the Hellenic alphabet. The recognition of these characters can be done with the use of a secondary OCR system of Hellenic characters. For the time being, our OBMR system is dedicated only to the recognition of the symbols of Table 1.
- One can easily observe that many characters are identical with others, and are distinguished only by their relative rotation of 45°, 90°, 135°, or 180° (e.g., the symbols "petasti" and "elafro" differ by a relative 180° rotation).

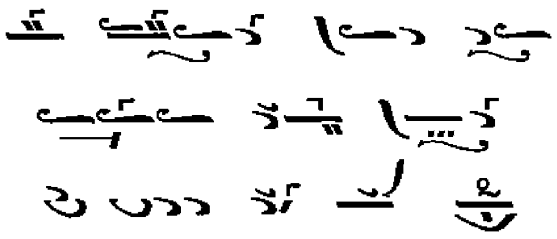


Fig. 2. Twelve character groups of the BMN.

- Moreover, as we can see from Fig. 1 and Table 1, there are characters (e.g., "clasma" and "oligon") with a substantial difference in size.
- Finally, each one of the resulting semantic musical groups consists of 2–10 and even more characters, and corresponds to a different note, or is performed by a *chanter* with a special musical way. This characteristic led us to the conclusion that an optical reader of the BMN should extract and recognize from left to right semantic musical groups of characters. Examples of such groups are given in the Fig. 2.

3. The OBMR system description

The developed OBMR system belongs to the category of the Off-line systems. The overall structure is shown in Fig. 3. First, the document is scanned at 300 dpi, which is a satisfactory resolution for OCR systems. Then, processing is divided in the following three stages:

- The *Segmentation stage*. The input is the scanned page and the stage is divided into three tasks. The first one is to segment the whole page into line pairs (the chant and the Hellenic syllables below it). The second one is to separate the chant from the Hellenic script below it. The third one is to extract the individual characters from the chant [13]. In the following, we consider that the three segmentation tasks have been executed and we have as outputs the bitmaps of the extracted characters. So far, standard segmentation techniques [14,15] have been adopted, and will not be discussed in more detail.
- The *Recognition stage*. It takes as input bitmaps of the characters of BMN, and consists of three main steps. The first one is the *preprocessing*, the second one is the *feature generation*, and the third one is the *classification*. We will discuss these main steps of the recognition process in detail in the next section. The output of the recognition stage is the corresponding character identification numbers (*cids*).
- The *Semantic musical group recognition stage*. It takes as input the character identification numbers (*cids*), as well as information from the segmentation stage related to the topological relationship of the characters (e.g. up, down, left, right, up-left, up-right, down-left, down-right, diagonal) [16]. Based on this information, we construct the semantic musical groups and recognize them using a database that works as a grammar and contains all the possible groups of the BMN (more than 2500). The output of this component would be the group identification numbers (*gids*).
- Finally, we have the *post processing step*, which takes as input the group identification numbers and gives as output the final result which may be the conversion to a true type font, or the performance of the BM.

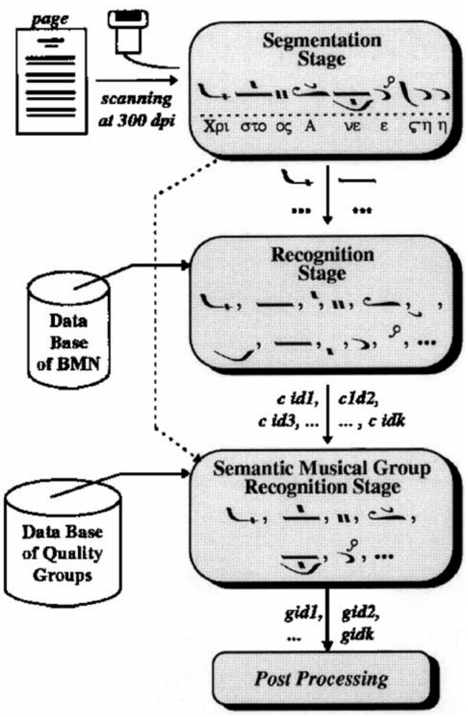


Fig. 3. The structure of the OBMR system in abstract form.

3.1. Description of the database of the BMN

The data set of the BMN that was used in the experiments of our system consists of about 18,000 symbols, in the form of 256 gray scale window bitmap files (*.bmp*) of uncompressed format. The database is about 71 MB in size. There are 71 classes, and for each class we have created 250 different patterns.

(i) 15–20% of the 250 pattern characters of each class are real scanned bitmaps from Byzantine Music books of different writing styles. We tried to scan different variations, for the same characters, so that to collect many printed styles for each character. Fig. 3a gives an example of different printed writing styles of the character "*ison*".

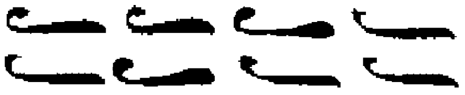


Fig. 3a. Different variations of the printed character “ison”.

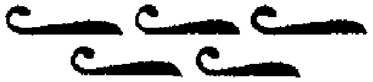


Fig. 3b. Character samples of class “ison” written by the Byzwriter 1.0 software.

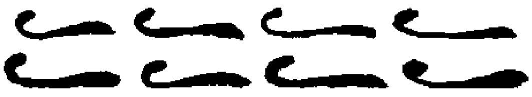


Fig. 3c. Character samples of class “ison” written carefully by hand.

(ii) Approximately 13% of the remaining 250 samples are characters that have been printed and scanned using a new software for writing the BMN in a computer, called Byzwriter 1.0 (developed by the author). The samples were written in the computer and were printed. Then, the printed samples were scanned. An example of this type of samples is shown in Fig. 3b.

(iii) Finally, the remaining bitmap samples of each class, which constitute the majority of the 250 patterns, were painted by hand carefully so as to resemble to the printed ones and also to have some kind of variation. In Fig. 3c examples of such painted characters are given.

Thus, we have created a data set that is semi-printed and semi-handwritten. Using this data set, the goal is to develop a system that recognizes printed fonts with high percentage. Furthermore, as we will see in our experimental results, using the semi-handwritten data set not only we obtain a high accuracy for the printed characters of BMN, but also good results are obtained for the recognition of handwritten BMN. In Fig. 3d we show some of the representative classes of characters in our database.

4. The recognition stage

The three main steps of the recognition stage are the following:

4.1. Preprocessing

The input in the preprocessing step consists of the bitmaps of BMN characters, being in 256 8-bit, gray scale window bitmap of uncompressed format. Having read the bitmap we follow the next preprocessing steps:

Thresholding: We binarize into black and white the gray scale bitmap using the algorithm in Ref. [17], which makes an automatic threshold selection derived from the gray-level histogram. An optimal threshold is selected by the gray-level histogram of the image, using its zero and the first moment. Other methods for binarization of a gray level image are given in Ref. [18].

Types of noise: The binarized character sample images can be affected by either of the following types of noise:

(i) The appearance of spurious dots or holes of one, two or more pixels in the image. This noise usually comes from a bad quality text or the binarization method. (ii) Noise may appear as spurious dots on the boundary of the shape. This causes a kind of variation to the character shape. (iii) Due to the handwriting of a character, the character boundary line may exhibit large variations. (iv) Another type of noise is the elimination of one or more internal holes, that are generic to the character, due to bad scanning, binarization, or handwriting. (v) Finally, another type of noise, is the faulty appearance of one or two internal character holes, due to scanning or binarization. The effects of these five types of noise are shown in Fig. 4a. In the following, we describe ways that eliminate the effects of some types of noise.

Dot/hole elimination & edge smoothing: This method is used to attack the first and second type of noise, which consists of spurious dots or holes, of one or two pixels, in the image or on the boundary line. For the elimination of these dots and holes we used the algorithm presented in Ref. [19]. According to this method, when a 3×3 window matches the pattern of Figs. 4b.a and 4b.b the central pixel is filled (hole elimination). Filling is also carried out when the window matches the rotated pattern (4b.b) in 90°, 180°, and 270°. On the other hand, deletion of the central pixel is carried out when a 3×3 window of the bitmap character matches the patterns (4b.c) and (4b.d) as well as the six equivalent patterns obtained from 90°, 180° and 270° rotations (dot elimination).

Using the above method we managed also to make a smoothing of the boundary shape of the character, so that, to eliminate the noise which overlays it. The other types of noise cannot be eliminated, and it is left to the robustness of the classification system to cope with.

Size normalization: This is the third step of preprocessing, in which the binarized character image is normalized into a fixed size. After extensive experimentation we concluded that the best image size for our OBMR system is 72×72. This is due to the following reason. If the size is greater than 72×72, we have in our image too many pixels, so this increases the complexity of the algorithms. On the other hand, if the size is chosen less than 72×72, then the normalized characters are very small, so that some of their structural features may be lost. We

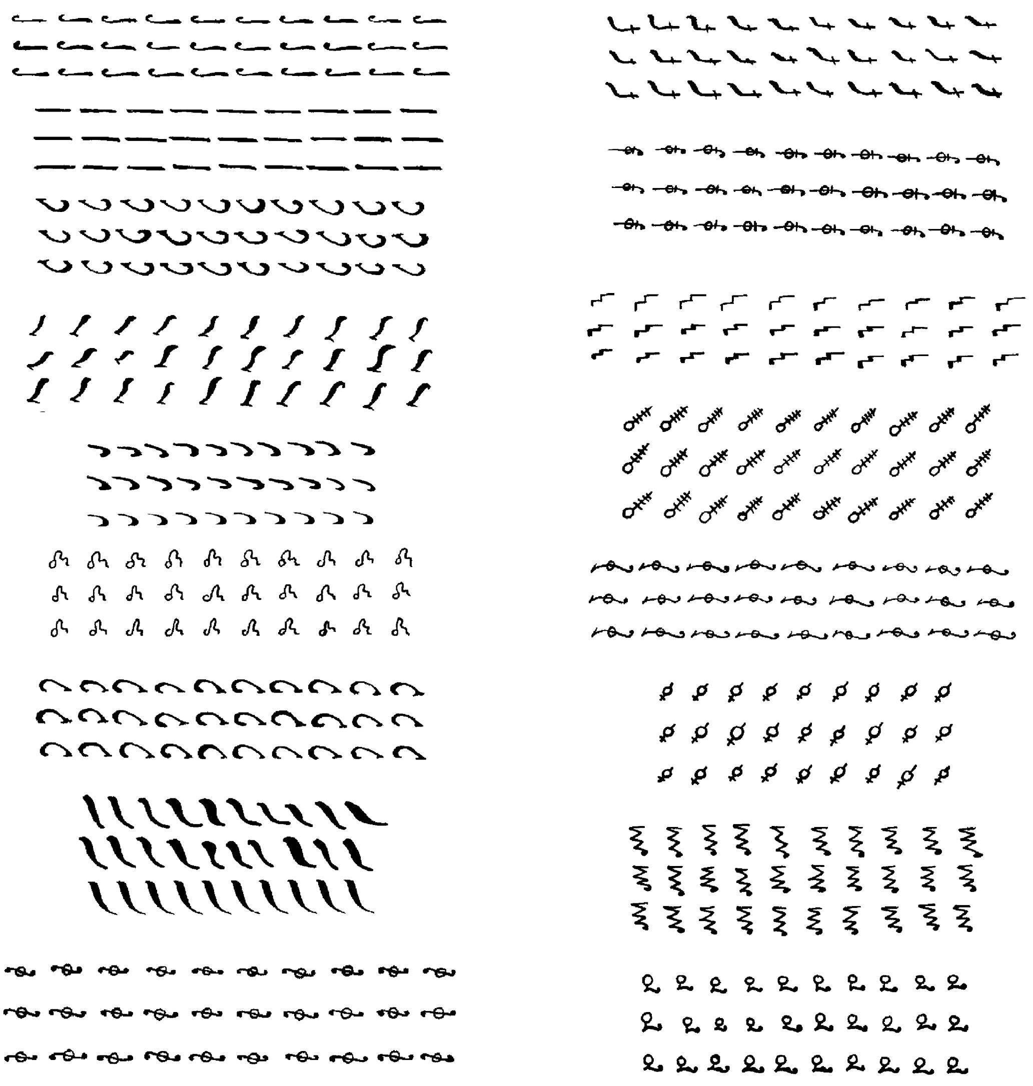


Fig. 3d. Samples of the BMN database.

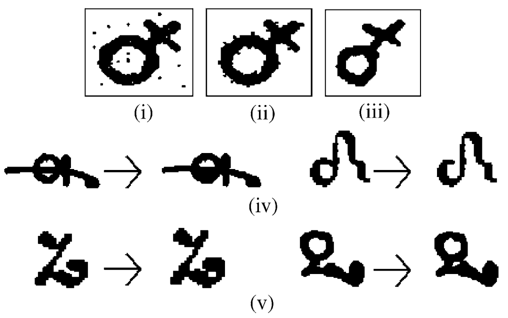


Fig. 4a. The five categories of noise, appeared in the character samples of the BMN.

?	M	?
M	o	M
?	M	?

(a)

?	M	?
M	o	M
o	o	M

(b)

o	o	?
o	M	M
o	o	M

(c)

o	o	o
o	M	M
o	o	?

(d)

Fig. 4b. The 3×3 patterns used for edge smoothing. Here "M" represents a black pixel. "o" represents a background pixel and "?" represents either.

developed a simple algorithm, which extracts the square bounding box of the character and scales the character to the 72×72 dimension. This makes the character images invariant to translations and scaling (see Fig. 4c) [8,13,20,21].

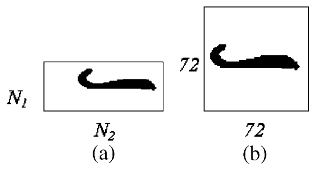


Fig. 4c. (a) The binarized character of dimension $N_1 \times N_2$. (b) The normalized character of dimension 72×72.

4.2. Feature generation

After normalization of the character and having made it invariant to translation and scaling, the character is passed into the feature generation stage.

At this point, it must be stated that, it is not desirable to make the characters invariant to rotations, as many OCR systems do, since many of our characters, as we mentioned earlier in Section 2.1, are identical and differ only with respect to their relative rotation.

Both structural and statistical features are generated [22]. The structural features are used during the preclassification stage and their statistical counterparts for the subsequent classification. The *structural features* used are the following:

Euler number: Euler's number E_N describes a simple topologically invariant property of the object. It is based on S, the number of contiguous parts of an object and H, the number of holes in the object. This number in our case corresponds to the number of contours of the character (external minus internal) or the number of holes, and is equal to

$$E_N = S - H$$

If we observe the set of the characters of Table 1, we can see that it is possible to preclassify them into *three main subsets*, using the structural feature of *Euler number*, since many of the characters have one hole, others have two holes, and others have none. We compute the Euler number using the algorithm presented in Ref. [13]. There are also other methods for the E_N computation, e.g., Refs. [21,23].

In our case, if $E_N = 1$, then we have the first subset of the characters without any hole (see Table 1). we have 39 such characters. If $E_N = 0$, then we have the second subset with 27 characters, having one hole and if $E_N = -1$, then we have the third subset with two holes and 5 characters.

However due to handwriting variation, or the binarization process (noise of type iv and v), it is possible for some of the characters without internal holes (or with one internal hole), to appear with one (or two) internal holes and v.v., (see Fig. 4a(iv) and (v)). So, these classes correspond to more than one Euler numbers (see Fig. 4d). The percentage of erroneous computation of E_N (due to noise) for the character samples of the first subset of 39 classes is 0.35%, for the second subset of 27 classes is 1.2% and for the third subset of 5 classes is 1.2%. To solve this problem, classes that may be affected by this type of noise, participate in more than one of the main subsets. For example, the character "delda" of Fig. 4d, will participate into the first main subset, if its $E_N = 1$, and into the second main subset, if its $E_N = 0$.

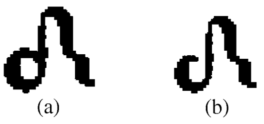


Fig. 4d. (a) The character "delta" with $E_N = 0$. (b) The character "delta" with $E_N = 1$, due to noise.

Principal axis: The principal axis of a bi-level object is the line passing through the object's center of mass having a minimum total distance from all pixels belonging to the object [13]. This is a relatively simple computation, and it can be made simpler by assuming that the line must also pass through one contour pixel. For the computation of the *PA* we compute first the external contour of the character. The algorithm we used for the external contour tracing will be described into the section of statistical features.

We compute also the *Center of mass* (C_x, C_y) of the image character using the zero and the first order moments M_{00}, M_{10}, and M_{01} from the equations as described in Refs. [13,21].

The *PA* is then computed as the line between the center of mass point C_M and the contour point $P(P_x, P_y)$, so that the sum of the distances of all the contour pixels from the line $C_M P$ to be minimum [13].

The general equation of a line passing through the two points is

$$ax + by + c = 0,$$

where

$$a = C_y - P_y, \ b = C_x - P_x$$

and

$$c = -(C_x - P_x) * C_y + (C_y - P_y) * C_x.$$

Then the distance between this line and a contour point (x_i, y_i) will be

$$d = \frac{ax_i + by_i + c}{\sqrt{a^2 + b^2}}.$$

The distance d is minimized for all contour points, in order to compute the *PA* and its direction ϑ. The direction of a line passing through two points (x_1, y_1) and (x_2, y_2)

is given by the formula:

$$\vartheta = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right).$$

Another algorithm for the computation of the *PA* direction ϑ is to use the second order central moments μ_{pq} from the equation [21,22]:

$$\vartheta = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right),$$

where central moments are given from

$$\mu_{pq} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i - C_x)^p(j - C_y)^q f(i,j).$$

Using the *Principal axis direction* we can also preclassify our symbols of Table 1, since subsets of them belong to distinct values of the *PA* direction, ϑ, (horizontal, vertical, right-diagonal, left-diagonal). After extensive experimentation, observing the direction of *PA* of all characters in the BMN database, we have reached the following preclasification schema:

```
If $E_N \neq 1$ (our character has one or two holes) then
    if $20^\circ \leqslant \vartheta \leqslant 72.9^\circ$ then slop = 1 (R-diagonal)
    if $72.9^\circ \prec \vartheta \leqslant 118^\circ$ then slop = 2 (vertical)
    if $118^\circ \prec \vartheta \leqslant 164.9^\circ$ then slop = 3 (L-diagonal)
    if $164.9^\circ \prec \vartheta \leqslant 72.9^\circ$ then slop = 0 (horizontal)
    else slop = 4,
else if $E_N = 1$ (our character has no holes) then
    if $10^\circ \prec \vartheta \leqslant 80^\circ$ then slop = 1 (R-diagonal)
    if $80^\circ \prec \vartheta \leqslant 121^\circ$ then slop = 2 (vertical)
    if $121^\circ \prec \vartheta \leqslant 163.5^\circ$ then slop = 3 (L-diagonal)
    if $163.5^\circ \prec \vartheta \leqslant 190^\circ$ then slop = 0 (horizontal)
    else slop = 4.
```

This is a preclassification schema based on Euler number and direction of *PA*. We quantized the space 0° to 360° into four main directions called *slops*. For the category of characters that contain one hole we have found, experimentally, the angles shown in the above pseudocode (including their supplementary angles) that define the four main slops. For characters that do not contain holes, we have found different values for the angles (including also their supplementary angles). Fig. 4e shows four characters with their *PA* and the four slops. Each class is associated with a slop, however due to *in-class variation* two or more *PA* directions are possible for some classes: For slop 0 the total number of classes is 20, while 4.6% of the respective character samples have a different slop. For slop 1, we have 24 classes and 3.22% of the respective samples belongs to a different slop. For slop 2 there are 12 classes and 14.2% of the samples have different slop, while for slop 3 we have 15 classes

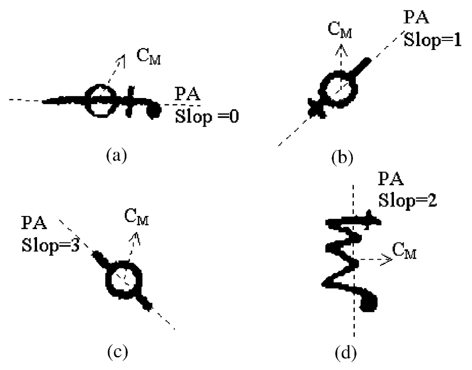


Fig. 4e. (a) The character "spathi" with Slop = 0 and $E_N = -1$, (b) the character "zygos" with slop = 1 and $E_N = 0$, (c) the character "fthora of Di" with slop = 3 and $E_N = 0$, (d) the character "fthora of Zo" with slop = 2 and $E_N = 1$.

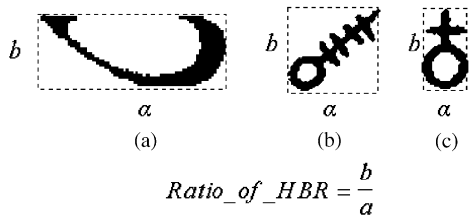


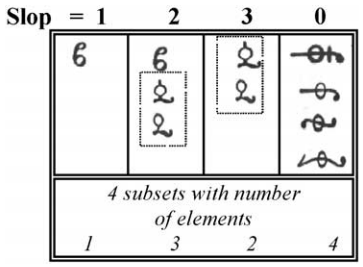
Fig. 4f. Horizontal Bounding Rectangle, b = height, α = width. If Ratio_of_HBR > 1 (c) then the character is extended vertically, if Ratio_of_HBR < 1 (a) is extended horizontally and if the Ratio_of_HBR≈ 1 (b) then the character is extended diagonally or is a square.

and 18.6% of their samples belongs to another slop category. To solve this problem we use the same procedure as in the case of E_N, allowing each class to participate in more than one slop categories (as it is described below), whenever it is needed.

Ratio of horizontal bounding rectangle: This feature is the ratio between the height and the width of the horizontal-bounding rectangle (*HBR*) of the image character. If instead of the horizontal, the minimum area bounding box is considered, then this feature is called Elogatedness [21]. If α is the width of the horizontal bounding rectangle and b the corresponding height, then we have the *Ratio_of_HBR* (see Fig. 4f). If this ratio is greater than 1, then the character is extended vertically (Fig. 4f(c)). On the other hand, if the ratio is less than 1, then the character is extended horizontally (Fig. 4f(a)) and if the ratio is approximately equal to 1, then the character is diagonal or square (Fig. 4f(b)).

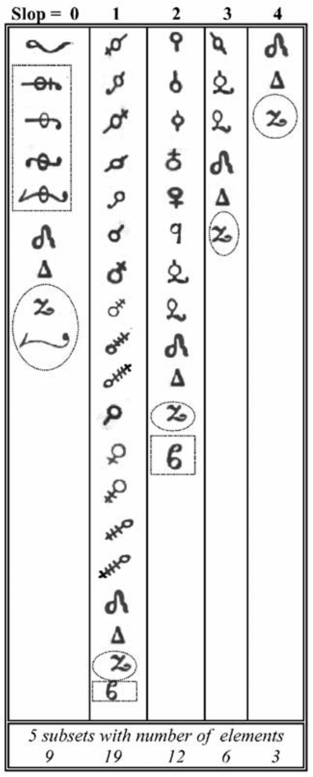
Using this feature we can also preclassify our character set of Table 1. Combining this feature with the *Euler Number* and the *PA direction*, we have the following tree-structured preclassification algorithm which

Table 2a
Subsets when $E_N = -1$ or -2



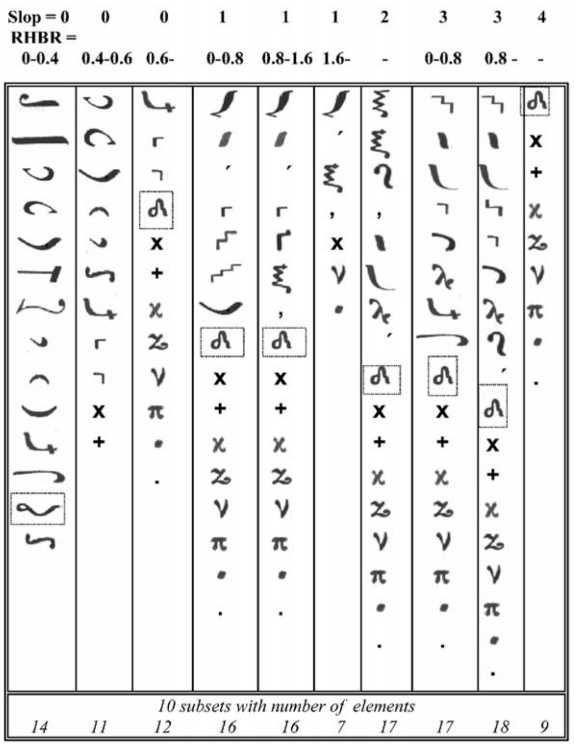
Slop =	1	2	3	0
4 subsets with number of elements	1	3	2	4

Table 2b
Subsets when $E_N = 0$



Slop =	0	1	2	3	4
5 subsets with number of elements	9	19	12	6	3

Table 2c
Subsets when $E_N = 1$



Slop =	0	0	0	1	1	1	2	3	3	4
RHBR =	0-0.4	0.4-0.6	0.6-	0-0.8	0.8-1.6	1.6-	-	0-0.8	0.8 -	-
10 subsets with number of elements	14	11	12	16	16	7	17	17	18	9

separates the set of 71 distinct BM characters into 19 subsets:

If $E_N = -1$ *or* -2, we have either two or three holes (due to noise) then the following 4 subsets are possible, depending on the slop feature (Table 2a). The characters in the dotted line box have usually $E_N = 0$, however due to noise, it is possible, in some cases, to result in $E_N = -1$.

If $E_N = 0$ (we have only one hole) then the following 5 subsets are possible depending on the slop feature (Table 2b). In this table, the characters in the dotted line box have usually $E_N = -1$, while the characters in the dotted line circle have usually $E_N = 1$, however, due to noise it is possible for these marked characters to give $E_N = 0$.

In addition to that, if $E_N = 1$, then we have 10 possible subsets, depending on the value of Slop as well as the Ratio of HBR. Table 2c shows, for example, the value of *slop* = 0 and the Ratio_of_HBR fluctuated from 0 to 0.4, for the first subset with 14 elements. The characters marked with a dotted line box, have usually $E_N = 0$, but due to noise, it is possible to have $E_N = 1$.

These three tables give us an hierarchical preclassification schema dividing the large set of 71 characters into 19 smaller subsets with a mean value of 10 characters per subset. This makes our classification algorithm simpler, faster and generally more efficient as we will see in the next section, since in the sequel we have to classify a character into one of at most 19 classes (characters) and not 71 (see Table 2b, second subset). This specific hierarchy is the result of extensive experimentation on the characters of the BMN database, together with a number of other possible choices. The adopted hierarchy is the one that gives the best results, since it exploits the specific characteristics of the Byzantine Music notation.

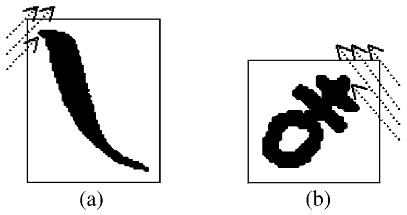


Fig. 4g. Adaptive Starting Point. (a) Left Diagonal Starting point (left diagonal scanning) if $E_N = 1$, (b) Right Diagonal Starting point (right diagonal scanning) if $E_N \neq 1$.

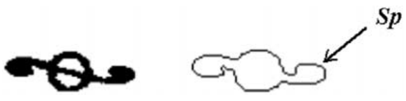


Fig. 4h. The character "fthora of Pa*", its contour and the Right diagonal starting point.

4.2.1. *Statistical features*

The statistical features that we have used are: (a) the discrete wavelet transform (DWT) of the contour function of the character and (b) the DWT of the 4-projections of the image character (horizontal, vertical, left-diagonal, right-diagonal). Let as now focus on the steps followed in the statistical feature generation:

Contour tracing: As many researchers in the area of OCR point out, the contour, which is defined as the outermost sequence of contour pixels in a bi-level image character, contains all the information that is necessary for the description of a character and its classification.

For the contour tracing of the BMN character set (see Fig. 4h), we developed an algorithm based on the technique given in Ref. [24]. In this algorithm we used an *adaptive starting point* depending on the E_N.

Adaptive starting point: The selection of the starting point into a contour tracing algorithm plays a significant role, since this may lead to increased variation of the contour, if it is not selected properly. This also causes more problems to the classification algorithm, if we use as features the DWT of the contour function, since the DWT is not invariant to the selection of the starting point [25].

To solve this problem we introduce the idea of an *adaptive starting point*. For our case, it turns out that adopting the value of the Euler number as the criterion for the choice of the starting point, results in a satisfactory performance. So, if $E_N = 1$, which means that the character has no holes inside, then we use the *Left diagonal starting point*, which is the first pixel that we meet if we scan the image character diagonally as it is pointed in Fig. 4g(a). On the other hand, if $E_N = 0$ *or* -1, which means that holes are present, then we use the *Right diagonal starting point*, which is the first pixel that we meet if we scan the image character as shown in Fig. 4g(b). The use of the adaptive starting point results in less variation of the contour function and better classification performance if we use discrete wavelet transform.

The contour can be represented as a closed parametric curve c in the complex plane C, i.e.,

$$c(i) = x(i) + jy(i), \quad i = 0 \ldots L - 1,$$

where L is the total number of points in the contour and j denotes the imaginary unit. We can consider that this closed curve is also a periodic function with period L. Then the DWT of c would be

$$DWT(c(i)) = DWT(x(i)) + jDWT(y(i)).$$

We know that the DWT requires the periodic function to have a period of L equal to a power of 2, if we want to decompose the signal up to the last level of decomposition [25,26]. However, it is very rare to have a contour with length equal to a power of 2.

Bezier splines of nth order: For this reason we developed an approximation method, which is based on *Bezier splines of nth order* [27,28], that can be fitted to any number of points. Using the L contour points, we can calculate, using the approximation method of Bezier splines, a new contour function which is an approximate version of the real contour and has 128 points length, which is a power of 2. We can also approximate our contour with $2^8 = 256$ points.

Using the Bezier curves of nth order, not only we construct a contour function with 128 points but we also achieve to have a smoother approximated version of the closed contour curve with reduced noise. The Bezier splines also give good results in cases where the characters are broken and the breaking points of the two parts of the character can be detected. In this case the Bezier splines have the property to compute the intermediate points of the contour at the breaking position. This is a way of elimination of the noise of the third category (see Section 4.1). As we increase the order n of the Bezier curves, a smoother version of the contour is obtained. We selected the order $n = 1$, which means that the approximated curve passes through the true contour points and all the contour information is preserved. An example of an approximated smoothed version of the contour is shown in Fig. 4i(c).

Using the approximated version of the Bezier curves the resulting vectors have a dimension of 128 points, that is $x(i),\ i = 0 \ldots 127$ and $y(i),\ i = 0 \ldots 127$, where x, y are the coordinates of the points of the new contour.

Adaptive projection method: Another feature results using the projection of the bi-level character into one of the four main directions. As it is pointed out in Ref. [20], the aim of the projection method is to simplify drastically a system of character recognition by reducing two

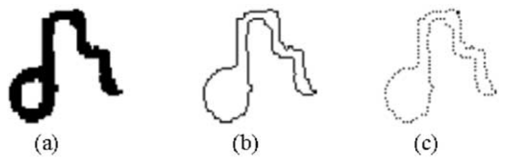


Fig. 4i. (a) The binarized and normalized version of the character "delta". (b) The real contour of length L. (c) The approximated smoothed version of the contour using the Bezier curves of first order with $128 = 2^7$ points.

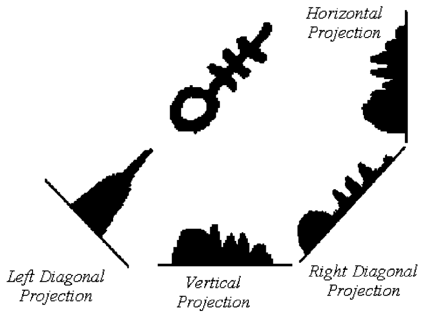


Fig. 4j. 4-Projections. The character "trigrami diesi" and its 4-projections (we filled the internal of the hole with black color for the projection).

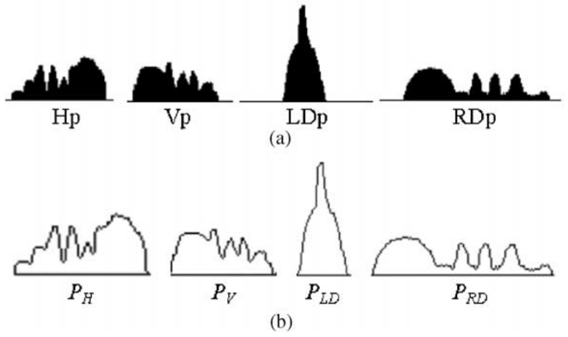


Fig. 4k. (a) The 4-projections of "trigrammi diesi". The lines between the projections corresponds to the zero value. (b) The four projection vectors P_H, P_V, P_{RD}, P_{LD}, without the zero values.

dimensional information to one dimension. Historically, this method appears in the early stage of OCR. The projection method is weak in diagonally oriented patterns in general.

What we have used in our OBMR system is the 4-projection method [10] adapted to the specific orientation each one of the character patterns, which overcomes the weaknesses associated with the diagonally oriented patterns. For each pattern an appropriate projection is used, depending on its direction.

For bi-level image characters we can calculate the 4-projections, i.e., the *horizontal*, the *vertical*, the *right-diagonal* and the *left-diagonal* (see Fig. 4j). The use of the right and left diagonal projection is very significant for our system, since we have many characters in this orientation. Using the 4-projections we manage to extract the geometrical structure of a character in the four directions and subsequently, to separate identical classes which differ only in their relative direction ϑ.

The projection in a direction, e.g., vertical, is defined as the function $Vp(i)$, $i = 0 \ldots N-1$, i.e., it is equal to the sum of the black pixels in this column of the image character.

For the computation of the horizontal Hp and the vertical projection Vp, we scan the image character from top to bottom and from left to right. However, for the computation of the diagonal projections LDp and RDp, we scan the image character diagonally as it is showed in Fig. 4g, used for the computation of the adaptive starting point.

The Vp and the Hp are of dimension $N = 72$, while the LDp and the RDp have dimension $2N - 1 = 143$ (Fig. 4k(a)).

Using the *adaptive projection method*, we extract four vectors P_H, P_V, P_{LD}, P_{RD}, which correspond to the non-zero values of the horizontal, vertical, left and right diagonal projections, respectively, as it is shown in Fig. 4k(b). Since our goal is to apply the discrete wavelet transform on these four vectors, the approximation method of *Bezier splines*, described in the previous section (Fig. 4i(c)), is also adopted for each one of the four projection vectors, in order the resulting vectors to have the same dimension of $128 = 2^7$ points. The result of this approximation is four new vectors $P_H(i), P_V(i), P_{LD}(i), P_{RD}(i)$, $i = 0 \ldots 127$. This was imposed by the goal to have enough levels for the subsequent wavelet decomposition as it is explained in the next section.

Discrete wavelet transform: Having formed six vectors, we will apply the discrete wavelet transform. The first one is $x(i)$, $i = 0 \ldots 127$ and constitutes the *x-coordinate* of the contour function, the second one is the vector $y(i)$, $i = 0 \ldots 127$, and constitutes the *y-coordinate* of the contour function and the remaining four vectors are the 4-*projections* $P_H(i), P_V(i), P_{LD}(i), P_{RD}(i)$, where $i = 0 \ldots 127$.

For the discrete decomposition of the DWT we have used Daubechies "db2" filters, with the following values for the low and high pass filters (see Fig. 4l).

We applied the wavelet decomposition onto the above six vectors, according to the algorithm given in Refs. [22,25,26,29]. The algorithm is as follows:

First the vector, $x(i)$, is passed through the low and high pass filters. The respective outputs are down sampled by 2, which means that from the vector of 128 points $(0 \ldots 127)$ we keep the values $(0, 2, 4, 6, \ldots, 126)$, i.e., 64 points. We keep the 64 points from the high pass and repeat the same procedure for the other 64 low passed

Low Pass = { -0.1294, 0.2241, 0.8365, 0.4830 }
High Pass = { -0.4830, 0.8365, -0.2241, -0.1294}

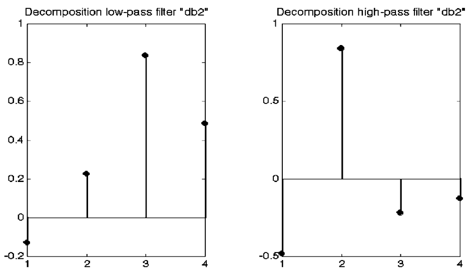


Fig. 4l. The Decomposition filter "db2" Daubechies 2. Low pass and High pass.

points. The above algorithm is repeated 7 times as one can see from Fig. 4m.

For the x vector of character contour we keep the lowest 16 wavelet descriptors which are: 1 *low pass* + 1 *high pass* + 2 *high pass* + 4 *high pass* + 8 *high pass*. That is, levels 4–7 of the decomposition are used [26].

The same schema is followed for the decomposition of the y contour vector of length 128, as well as for the 4-projection vectors $P_H(i), P_V(i), P_{LD}(i), P_{RD}(i)$, $i = 0 \dots 127$, where we keep the 16 lowest wavelet descriptors, as we did for the x contour vector.

Final feature vector: To obtain the final feature vector F, we apply the DWT on each one of the above six vectors. After extensive experimentation on the appropriate combination of contour and projection features, it was found that if we use both the wavelet descriptors of the contour and the wavelet descriptors of only one projection, that corresponds in the direction of the character slop, we obtain the best results, compared to all other combination possibilities. This will be further described in Section 5. The following algorithm gives the final feature vector, F, depending on the slop value of the character.

If slop = 0, then the *feature vector* consists of 48 coefficients = {the 16 wavelet descriptors of the x decomposition 16 wavelet descriptors of the y decomposition + 16 wavelet descriptors of the $P_V(i)$ decomposition (we use only the vertical projection)}.

If slop = 1, then the *feature vector* consists of 48 coefficients = {the 16 wavelet descriptors of the x decomposition 16 wavelet descriptors of the y decomposition + 16 wavelet descriptors of the $P_{RD}(i)$ decomposition (we use only the right diagonal projection)}.

If slop = 2, then the *feature vector* consists of 48 coefficients = {the 16 wavelet descriptors of the x decomposition 16 wavelet descriptors of the y decomposition + 16 wavelet descriptors of the $P_H(i)$ decomposition (we use only the horizontal projection)}.

If slop = 3, then the *feature vector* consists of 48 coefficients = {the 16 wavelet descriptors of the x decomposition + 16 wavelet descriptors of the y decomposition + 16 wavelet descriptors of the $P_{LD}(i)$ decomposition (we use only the left diagonal projection)}.

If slop = 4, then the *feature vector* consists of 32 coefficients = {the 16 wavelet descriptors of the x decomposition 16 wavelet descriptors of the y decomposition (we use only the contour decomposition)}.

At this point, it must be emphasized that in this type of generated features, i.e., the approximated version of the contour function (using Bezier splines), the computation of the 4-projections and their approximation with the Bezier splines and finally their DWT combinations, has not been used before. Although a number of other "off-the self" techniques was used, we concluded that our choice of the above feature vector serves the needs of our problem best. This is because, using the Bezier approximation method, we can drop out some of the variation of the contour function. On the other hand, adopting from the 4-projections, the one lying in the direction of the character slop, together with the contour function, we can describe the structure of the characters much better than if we had used the contour information only. With

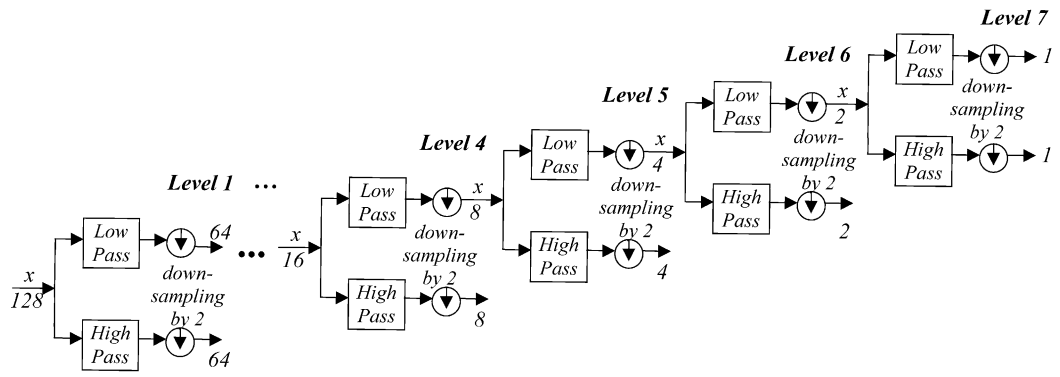


Fig. 4m. The discrete wavelet decomposition of vector x(i), where $i = 0 \dots 127$. The length of x is $128 = 2^7$, that corresponds to seven levels of decomposition.

this combination, we can separate similar characters better, as we will see with the quoted experimental results later on. Moreover, compressing this information using the discrete wavelet decomposition results in a final feature vector with low dimension.

4.2.2. *Classifier design*

For the final classification of the described feature vector the nearest-neighbor classifier was used, giving better results compared to other classifiers, such us neural networks. Experiments for the comparison of these classifiers will be presented in the Section 5.

The nearest neighbor classifier labels an unknown character, represented by the m-dimensional feature vector $F=[f_1, f_2, \ldots, f_m]$, with the label of the nearest neighbor of F among all the training patterns [30]. In our case, we have 71 classes and 250 patterns for each class. We used $n=50$ patterns from each class for the training of the NNC. This means that we have $71 \times 50 = 3550$ training patterns $T^c=[t^c_{k1}, t^c_{k2}, \ldots, t^c_{km}]$, where $m=48$, $k=1,\ldots,50$, $c=1,\ldots,71$.

Let F be an unknown test feature vector. Then F is classified into the class C^* if:

$$C^* = \min_{c=1,\ldots,71} d(F, T^c),$$

where d is the Euclidean distance between the test vector F and the vector T^c equal to

$$d=\sqrt{(f_1-t^c_{k1})^2+(f_2-t^c_{k2})^2+\cdots+(f_m-t^c_{km})^2}.$$

It has been suggested that, in order to prevent the domination of a subgroup of features in the feature vector, one can normalize the feature vectors T^c. The normalization consists of subtracting sample mean and dividing by standard deviation of the corresponding class c [30]. However, in our case we found that better results are obtained without normalization. This is because in some classes there are subclasses of the same character due to (a) the selection of the starting point and the contour extraction and (b) the variation within the class. This problem has also been pointed out in Ref. [31] (Fig. 4n). As we can see from Fig. 4n, the class "ipsili" may have two subclasses due to the variation in shape and due to different left diagonal starting point. This means that during the training phase of the NNC, for the character "ipsili", we must include into the set of the n representative training patterns all the subclasses of the character. Then, if we normalize our training patterns by subtracting the mean vector and dividing by the standard deviation vector we will have undesirable results, for the cases where subclasses exists, since the standard deviation will have large values due to the large differences in the representative training patterns.

So, for our system the Euclidean distance was used without normalization for the NNC, and we made provision to include all the subclasses for each class during the training phase.

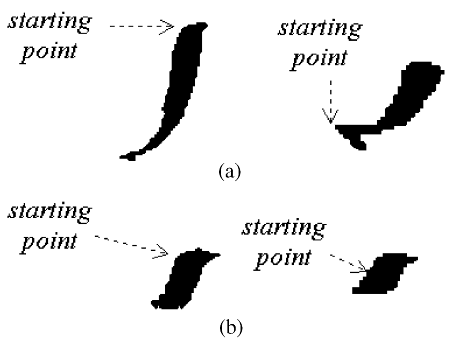


Fig. 4n. (a) The class "ipsili" with two subclasses coming from the variation of the shape and the different left diagonal starting point. (b) Two subclasses of "yporoi".

5. Experimental results

For our experiments we used the 18 000 symbols of the database of BMN. A large number of experiments were carried out, where the classifier was trained with 50 out of the 250 patterns of each class and tested with the remaining 200 patterns. So, 3550 training patterns and 14 200 testing patterns were used. Using a smaller number of training patterns than testing ones and obtaining good results, indicates that the adopted feature vector is well conceived and the system has good generalization ability.

After extensive experimentation we concluded that the preclassification schema, described in the former sections, gave the best results. Thus, we used all the structural features and the preclassification schema given in Tables 2a–2c and the following experiments were carried out:

(i) A feature vector of dimension equal to 32 was used containing the 16 wavelet descriptors of x-contour coordinate + 16 wavelet descriptors of the y-contour coordinate. With this experiment we tested the accuracy of the system if only the contour information of the characters is used. We used the NNC, trained with 50 patterns from each class and made a test with the remaining 200 patterns. For those of the classes that contain subclasses, we used different representative sets of the 50 training patterns, in order to include all the subclasses into the training set. The results of Table 3a have been obtained. In this experiment an average accuracy of 97.2% was achieved.

(ii) A feature vector of dimension equal to 64 was used containing the 16 wavelet descriptors of x-contour coordinate + 16 wavelet descriptors of the y-contour coordinate + the 32 wavelet descriptors (resulted from a 7-level decomposition) of the vector that contains the

Table 3a
The 71 symbols of Table 1 and the respective recognition accuracies using only the contour information

id	(%)	*id*	(%)	*id*	(%)	*id*	(%)
1	99.5	19	98.5	37	95.5	55	94
2	100	20	100	38	90.5	56	99
3	100	21	100	39	97.5	57	91.5
4	99	22	99.5	40	97.5	58	97
5	97	23	96.5	41	96.5	59	100
6	99.5	24	99.5	42	98	60	97.5
7	98.5	25	99	43	98.5	61	99
8	99.5	26	99.5	44	99.5	62	100
9	99	27	99	45	100	63	100
10	99.5	28	97.5	46	100	64	99.5
11	93	29	98.5	47	99.5	65	100
12	100	30	98.5	48	91	66	98.5
13	100	31	95.5	49	72.5	67	100
14	96	32	100	50	88.5	68	99
15	100	33	94.5	51	100	69	98
16	97	34	100	52	95.5	70	96.5
17	95	35	100	53	90	71	95
18	99.5	36	99.5	54	78.5		
Average accuracy= 97.2%							

Table 3b
Experiments for "trigrammi yfesi", *N* is the feature vector length, "16wtX" is the 16 wavelet descriptors of vector X. The best results are obtained using the contour information and the P_{RD}, 86%

N	Features for character "trigrammi yfesi", " "	(%)
96	16wt X+16wt Y+16wt P_V+16wt P_H +16wt P_{LD} +16wt P_{RD}	74.5
48	16wt X+16wt Y+16wt P_V	69
48	16wt X+16wt Y+16wt P_H	67
48	16wt X+16wt Y+16wt P_{LD}	61
48	16wt X+16wt Y+16wt P_{RD}	86
64	16wt X+16wt Y+16wt P_V+16wt P_{RD}	76.5
32	16wt X+16wt Y	78.5
16	16wt P_{RD}	81

Table 3c
The 71 symbols of Table 1 and the respective recognition accuracies using the contour information together with the projection lying in the direction of the character slop. A better accuracy of 98.1% was achieved

id	(%)	*id*	(%)	*id*	(%)	*id*	(%)
1	100	19	97.5	37	98.5	55	96
2	100	20	100	38	86	56	99
3	100	21	100	39	98.5	57	92.5
4	99	22	100	40	96.5	58	97
5	97	23	99	41	98.5	59	100
6	99	24	99	42	99	60	98.5
7	99	25	100	43	100	61	98.5
8	99.5	26	100	44	99.5	62	99.5
9	99	27	100	45	99.5	63	100
10	99.5	28	96	46	100	64	98.5
11	92.5	29	99.5	47	99	65	100
12	100	30	98.5	48	96	66	99.5
13	100	31	95.5	49	92.5	67	100
14	96	32	99.5	50	94.5	68	99.5
15	99.5	33	98	51	100	69	99
16	99.5	34	100	52	97	70	96
17	96	35	100	53	94	71	96.5
18	99	36	99.5	54	86		
Average accuracy = 98.1%							

4-projections approximated to 128 points using B-Splines method. This experiment tested the accuracy of the system if the contour information is used together with 4 projections of the characters. We used the same training and testing scheme as in the previous experiment. This experiment gave similar results as before with an accuracy of 97.1%.

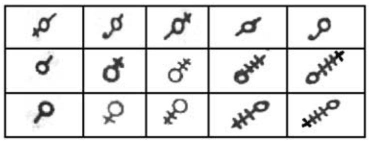
(iii) In contrast to the previous experiment we found that combining the contour information together with the projection determined by the direction of the corresponding character slop, improves the performance substantially.

For example, the recognition accuracy of the character "trigrammi yfesi" with slop = 1 (right diagonal) from 62% increased to 86% if the contour information is combined with the P_{RD} only (right diagonal projection) for the wavelet decomposition (see Table 3b). We can also observe that for this case, using only the right diagonal projection information, dropping out the contour information, we obtain better results (81%) than that obtained from the contour information only (78.5%). In other cases, for other classes, using only the contour information can lead to better results. However, the combination of the contour information with that of the projection lying in the direction of the character slop gives the best results.

Thus, the adopted feature vector was the one that combines the contour information with only one projection, depending on the character slop. The results of Table 3c have been obtained. The same training and testing scheme as in the previous experiment was used. In this experiment an average accuracy of 98.1% was achieved.

(iv) The goal of this experiment was to use a Neural Network classifier and see if a performance gain can be achieved compared to that of the nearest neighbor classifier. Emphasis was put on those classes where the nearest neighbor classifier resulted in relatively low performance. Let us, for example, focus on 15 classes, where the symbols are very similar and the classification is a

Table 3d
A set of 15 classes having many similarities



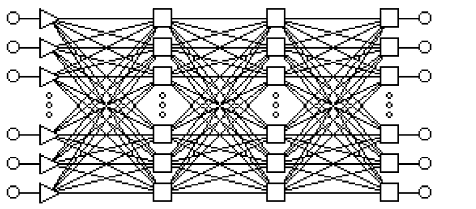


Fig. 5a. A Multilayer Perceptron with two hidden layers, 1 input with 48 nodes and 1 output with 15 nodes.

difficult task. This set of classes is shown in Table 3d. The average recognition accuracy for this set, according to the Table 3c when the nearest neighbor classifier is used, is 96.4%.

A Multilayer Perceptron (MLP) with 2 Hidden layers was used. The input layer contained 48 nodes, and the same feature vector was used as with the Nearest Neighbor Classifier. The output layer consisted of 15 nodes, where each node corresponds to a class (see Fig. 5a). For each neuron the log-sigmoid transfer function was used. For the training the back-propagation algorithm was adopted [22,32], in the form of the *scaled conjugate gradient algorithm* (*SCG*) presented in Ref. [33], having the learning rate was equal to 0.05.

Many experiments were carried out in order to estimate the best number of nodes for the hidden layers of the network. Initially, a pruning algorithm was used, e.g., Refs. [22,32], in order to get an estimate of the range where the number of nodes lie. Using this as a base, the number of nodes varied in order to obtain the best performance. Table 3e summarizes the results. The training of the network was terminated when the training error was of the order of 10^{-5}. The task of possible overfitting was also considered and training was terminated when the error was of the order of 10^{-3}. However, always this training resulted in higher error probability estimates. As it is apparent from Table 3e, the nearest neighbor resulted in slightly better performance. This trend was also the same when the number of training patterns was increased to 100 and the number of test patterns was reduced to 150, for each class. Similar comparative results were obtained for all the subsets resulting from the original tree structure.

Table 3e
The network architecture input/first HL/second HL/output, the number of epochs where the network error is less than 10^{-5}, and the network accuracy

Network architecture	Epochs	(%)
48/150/100/15	365	95.20
48/100/50/15	628	95.43
48/55/35/15	1069	94.53
48/47/30/15	1345	93.66

5.1. A post-classification scheme

A close observation of Table 3c reveals that some of the symbols are recognized with low accuracy (e.g., the symbols 48–50 and 53–55). This is because they are confused with other similar symbols. These can be grouped in the following cases (see Fig. 5b): (a) the symbols "fthora of Vu" and "fthora of Zo" differ only in a vertical small line appearing in the second symbol. In case (b) we have the symbols "fthora of Ni" and "fthora of Ni*" that differ also in a small vertical line present in the second character. In case (c) we have three symbols, i.e., "zygos", "fthora of Ni**" and "fthora of Di**" that differ only in a diagonal small line. It's obvious that all these are very similar. In the case of (d) we have three characters where they differ in the number of the diagonal lines (i.e., 2, 3, 4 diagonal lines). The same confusion occurs in case (e) where the characters are similar and differ only in the number of the diagonal lines. In case (f) we have the characters "spathi" and "fthora of Di*" that differ only in a vertical small line and in the case of (g) we have the characters "gorgo" and "ga" that are very similar. In case (h) we have the "digorgo" and "trigorgo" that differ only in one a small step in their shape. In case (j) we have "apli" and "stigmi" which are too small, resemble to a dot and one matches to a square where the other to a small circle.

In order to solve these confusions we developed one algorithm, for each case, that is based on structural features, generated from the confused characters, e.g. Ref. [34]. In the sequel, we briefly present the algorithms that exploit the structural characteristics and the particularities of these confused characters.

For the confusion set of Fig. 5b(a), we developed an algorithm which computes the three points P_1, P_2 and P_3, on the upper part of the two characters shown in Fig. 5c, where P_2 is the highest point between the edge points P_1 and P_3. Then we compute the distance d between the point P_2 and the line segment P_1P_3. If $d > T_d$, where T_d is predefined threshold value, then the

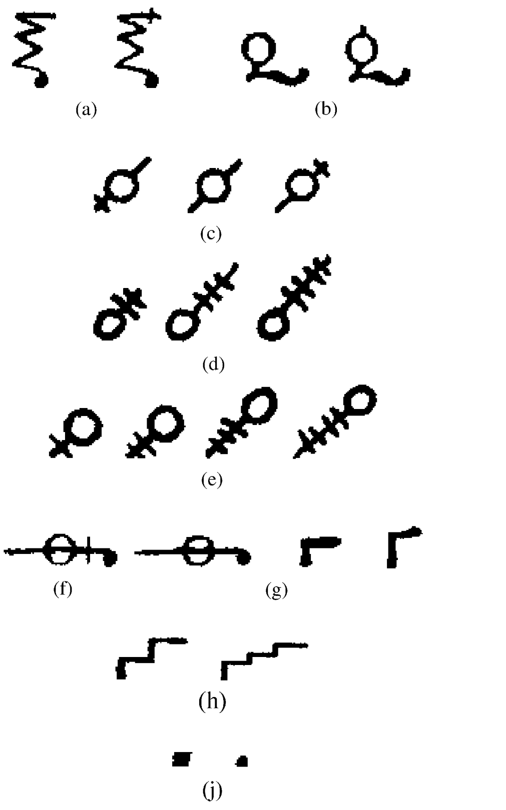


Fig. 5b. The confusion set of BMN according to the classification schema we described that reduces the total system accuracy. The characters in each subgroup (a–j) are confused each other using the wavelet descriptors.

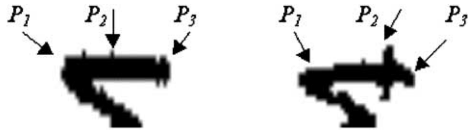


Fig. 5c. The algorithm that solves the confusion of Fig. 5b(a) The upper part of the two characters.

character is the "fthora of Zo", otherwise the character is the "fthora of Vu".

For the confusion set of Fig. 5b(b), we developed another algorithm which computes the highest point of the upper part of the two characters, $P_{\max}$, and two other equidistant points left and right of $P_{\max}$, P_1 and P_2, shown in Fig. 5d. Then if the angle θ between the line segments $P_1P_{\max}$ and $P_2P_{\max}$ is greater than a predefined threshold T_θ, then the character is the "fthora of Ni", otherwise it is the "fthora of Ni*".

For the character confusion set of Fig. 5b(c), we developed a different algorithm that estimates the line which

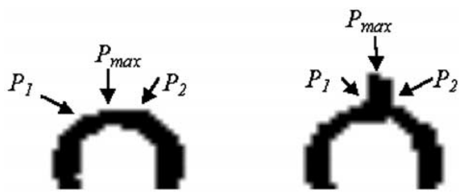


Fig. 5d. The algorithm that solves the confusion of Fig. 5b(b). The upper part of the two characters.

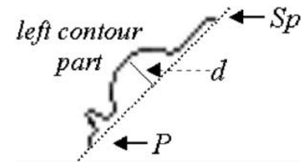


Fig. 5e. The algorithm that solves the confusion of Fig. 5b(c). The left part of the contour of character "zygos" and the line between the points Sp and P.

passes from the right diagonal starting point of the contour, Sp, and the contour point P corresponding to the maximum distance from Sp. These two points separate the contour into two parts. The left part is shown in Fig. 5e. We created a vector C_A that contains the distances d of the contour pixels of the left part, from the line SpP and a vector C_B for the corresponding right part. A smoothing procedure was applied on these vectors, using a two by two averaging. This smoothing is applied several times and it turns out that the resulting vectors consist of only very small and very large values. This is very useful for using thresholds. Then we search if there are large values in the 8 first and 8 last positions of the vectors. If there are large values in the 8 last positions of the vectors then we have the character "zygos" (the first one in Fig. 5b(c)). If there are large values in the first 8 positions then we have the character "fthora of Di**" (the second one in Fig. 5b(c)). Finally, if no large values occur in the first and last positions of the vectors, then we have the character "fthora of Ni**" (the third one in Fig. 5b(c)). This algorithm exploits the symmetric form of the three characters as well as their minor differences.

For the character set of Fig. 5b(d) and 5b(e), we used the same algorithm since these characters are similar and differ only by a relative 180° rotation. The algorithm follows the same philosophy of the previously described algorithm. The only difference is that in this algorithm we search the smoothed vector C_A and C_B and count the number of maximum and minimum values. Let k_1 be the number of maximum values in C_A and k_2 the corresponding number of minimum values, and also k_3 and k_4 are the corresponding minimum and maximum values of the right part C_B. Then we sum up $k = k_1 + k_2 + k_3 + k_4$ and if k is greater or smaller than predefined

Table 3f
Experiment employing the post classification schema. The final accuracy is 99.1%

id	(%)	*id*	(%)	*id*	(%)	*id*	(%)
1	100	19	100	37	100	55	100
2	100	20	100	38	100	56	99
3	100	21	100	39	99.5	57	92.5
4	99	22	100	40	98	58	97
5	97	23	99	41	98.5	59	100
6	99	24	99	42	99	60	98.5
7	99	25	100	43	100	61	98.5
8	99.5	26	100	44	99.5	62	100
9	99	27	100	45	99.5	63	100
10	99.5	28	98.5	46	100	64	98.5
11	92.5	29	99.5	47	99	65	100
12	100	30	99	48	97	66	99.5
13	100	31	100	49	99	67	100
14	96	32	99.5	50	100	68	99.5
15	99.5	33	100	51	100	69	99
16	99.5	34	100	52	98.5	70	96
17	99.5	35	100	53	99	71	96.5
18	99.5	36	99.5	54	100		
Average accuracy = 99.1%							

thresholds, we can separate the three characters of Fig. 5b(d) and the ones of Fig. 5b(e).

Concerning the set of Fig. 5b(f), we used the same algorithm described for those of Fig. 5b(c). For the cases of 5b(g) and 5b(h) we make a second classification using only the contour information for the final feature vector. Finally, the separation of "apli" and "stigmi" (Fig. 5b(j)) can be done easily during the semantic musical group recognition phase, since the two characters are always placed in different semantic positions in the groups.

Repeating the former experiment (iii) with the same characteristics and employing the post classification part, which overcomes the "confusion set" of Fig. 5b, we obtain an average accuracy of 99.1% shown in Table 3f.

If we compare Tables 3c and 3f in the positions of confused symbols we observe the large improvement in the accuracy.

5.2. Final evaluation

For the final evaluation of the system the cross validation method (Leave-ten-out-method) was adopted. According to this method 90% of our patterns were used as training patterns, for the NNC, and the remaining 10% for testing. The cross validation method resulted in the final accuracy of 99.4%.

Finally, the described method was applied on real scanned pages of Byzantine music text. First the page with the printed Byzantine psalm of Fig. 1 "Christ is Risen" was fed into the system. The recognition resulted in 100% accuracy. The recognition accuracy of others

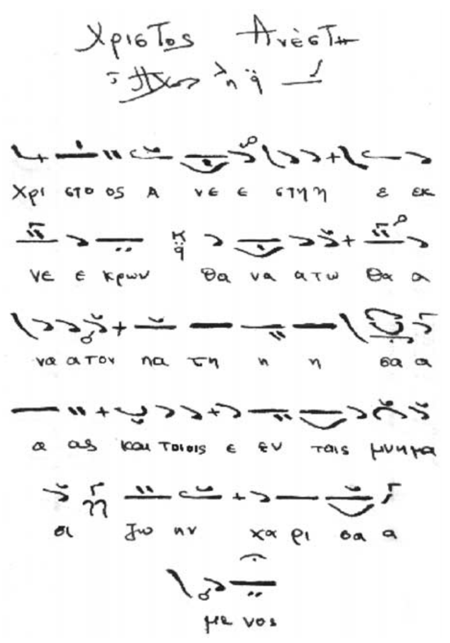


Fig. 5f. The psalm "Christ is Risen" in carefully handwritten form and recognition accuracy 92.85%.

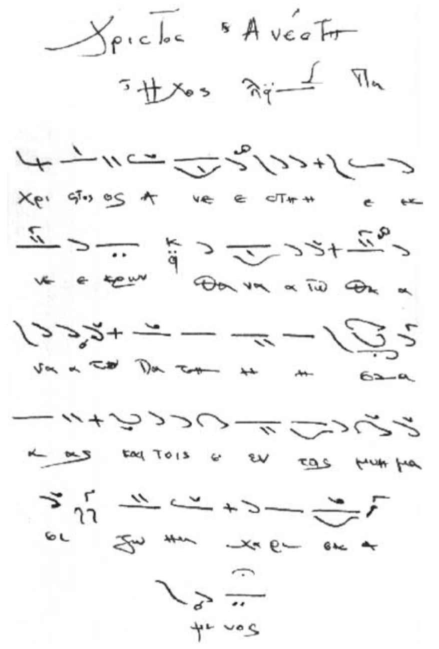


Fig. 5g. The psalm "Christ is Risen" in cursive handwritten script and recognition accuracy 73.2%.

psalms, scanned from Byzantine music books was ranged from 96% to 99.5%.

To test the robustness of our system, for the case of handwriting Byzantine text recognition, we wrote the same psalm of "Christ is Risen" in two different forms. The first one was written carefully, so as to resemble to a printed one, Fig. 5f. The second one was written very quickly with single strokes and large variation (cursive script), Fig. 5g. The results were 92.85% recognition for Fig. 5f and 73.2% for Fig. 5g. These results indicate the robustness of our system in recognizing the printed Byzantine music text as well as the carefully handwritten text.

6. Conclusion

Although OCR research appeared in the area of western or classical music in the decade of 1960, it is the first time that an OCR system is developed for the Byzantine Music notation. In this paper we presented a first approach for an OBMR (optical Byzantine Music recognition) system that is able to optically recognize the 71 distinct characters of the BMN.

A tree-structured classification schema was adopted based on (a) the structural features of the Euler number, the Principal axis direction and the ratio of the horizontal bounding rectangle of the character and (b) on statistical features generated from the approximated version of the contour of each character. For this approximation we used the *n*th order Bezier splines. The 4-projections of the characters (horizontal, vertical, left diagonal, right diagonal) were computed and then the DWT (discrete wavelet transform) was applied on the contour function and the projection vector lying in the direction of the character slop, in order to form feature vectors. The classifiers used were the nearest neighbor and the multilayer perceptron, with the former giving better results. The overall accuracy of the system, employing a post classification scheme, to overcome confusions, was estimated using the cross validation method to be 99.4%.

Currently, more advanced techniques are under consideration, such as deformable template matching, Vector support machines, and also a post processing grammar is being developed for the recognition of the BMN as semantic musical groups.

7. Summary

In this paper, a new OCR system is presented for the first time. This is an optical Byzantine Music recognition (OBMR) system, which recognizes the printed notation of the orthodox Hellenic Byzantine Music that has been established for use since 1814.

Byzantine Music is a special type of phonetic music that has been developed, formed and cultured, within the Hellenic Orthodox Church. Byzantine Music has its own notation (BMN) and its own way of performance. The notation of the Byzantine music consists of 71 distinct symbols (characters), which are combined to form group of symbols, each one having its own semantic musical meaning and its own musical performance.

For the needs of the system a database of approximately 18000 bitmaps of scanned, printed and handwritten Byzantine musical characters was created. For the recognition of the characters both structural and statistical features were generated. The structural features are: (i) the Euler number (ii) the principal axis direction and (iii) the ratio of the horizontal bounding rectangle of the character. These structural features were used in a hierarchical preclassification schema that divides the large set of 71 character classes, into 19 smaller subsets. This specific hierarchy was the result of extensive experimentation and exploits the specific characteristics of the BMN and it leads to an efficient classification system.

The generated statistical features were: (i) the discrete wavelet transform (DWT) applied onto the coordinate vectors x and y of the approximated version of the contour function of the character. For the approximation of the contour the Bezier splines method was used. Also, the idea of an adaptive starting point of the contour was adopted (ii) The DWT applied onto the horizontal, vertical, left-diagonal and right-diagonal projection vectors and the selection of the projection lying in the direction of the character slop. The final feature vector, used for the classification, consisted of the16 lowest resolution wavelet descriptors of the x contour co-ordinate, the 16 lowest resolution wavelet descriptors of the y contour co-ordinate and the 16 lowest resolution wavelet descriptors of the selected projection vector. The classifiers used were the nearest neighbor and the multilayer perceptron, with the former giving better results.

Experiments, using the above classification schema and features, resulted to an average error rate of 98%. However, some of the characters were mutually confused. A post classification stage, taking care for these confusions, was developed exploiting specific structural characteristics of the involved characters. This further increased the recognition accuracy to 99%. This performance evaluation was carried out using the cross validation method. Finally, the recognition ability of the system was tested with a number of printed Byzantine musical texts, and the recognition accuracy was ranged from 96% to 100%.

References

[1] D.G. Panagiotopoulos, Theory and Practice of the Church Byzantine Music, 1991.

[2] K.A. Psachos, The Parasimantiki of the Byzantine Music, Dionysos Publishing Co, Athens, 1978.
[3] D. Pruslin, Automatic recognition of sheet music, Sc. D. Dissertation, Massachusetts Institute of Technology, 1966.
[4] D.S. Prerau, Computer pattern recognition of standard engraved music notation, Ph.D. Thesis, Massachusetts Institute of Technology, 1970.
[5] D.S. Prerau, Computer pattern recognition on printed music, in Proceedings of the Fall Joint Computer Conf. Montvale, NJ, 1971, pp. 153–162.
[6] D.S. Prerau, Do–Re–Mi: a program that recognizes music notation, Comput. Humanities 9 (1975) 25–29.
[7] D. Bainbridge, N. Carter, Automatic reading of music notation, Handbook of Character Recognition and Document Image Analysis, 1997, pp. 583– 603.
[8] H. Bunke, P.S.P. Wang, Handbook of Character Recognition and Document Image Analysis, World Scientific, Singapore, 1997.
[9] V.G. Gezerlis, S. Theodoridis, An optical music recognition system for the notation of the orthodox hellenic Byzantine music, Proceedings of ICPR2000 conference, Barcelona, Sept. 2000.
[10] T.H. Hilderbrandt, W. Liu, Optical recognition of handwritten chinese characters: advances since 1980, Pattern Recognition 26 (2) (1993) 205–255.
[11] A. Amin, Off-line arabic character recognition: the state of the art, Pattern Recognition 31 (5) (1998) 517–530.
[12] Øivind D. Trier, A.K. Jain, T. Taxt, Feature extraction methods for character recognition-a survey, Pattern Recognition 29 (4) (1996) 641–662.
[13] J.R. Parker, Practical Computer Vision Using C, Wiley, New York, 1994.
[14] R.G. Casey, E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. PAMI 18 (7) (1996) 690–706.
[15] Y. LU, M. Shridhar, Character segmentation in handwritten words—an overview, Pattern Recognition 29 (1) (1996) 77–96.
[16] S.W. Lu, Y. Ren, C.Y. Suen, Hierarchical attributed graph representation and recognition of handwritten chinese characters, Pattern Recognition 24 (7) (1991) 617–632.
[17] N. Otsu, A threshold selection method from Gray–Leve histogram, IEEE Trans. System Man. Cybernet. SMC-9 (1) (1979) 62–66.
[18] Øivind Due Trier, A.K. Jain, Goal-directed evaluation of binarization methods, IEEE Trans. PAMI 17 (12) (1995) 1191–1201.
[19] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, L. Lam, Computer recognition of unconstrained handwritten numerals, Proc. IEEE 80 (7) (1992) 1162–1180.
[20] S. Mori, H. Nishida, H. Yamada, Optical Character Recognition, Wiley Series, New York, 1999.
[21] M. Sonka, V. Hlavac, R. Boyle, Image Processing, Analysis and Machine Vision, Chapman & Hall, London, 1995.
[22] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, New York, 1998.
[23] J.L. Diaz de Leons, J. Humberto Sossa-Azuela, On the computation of the Euler number of a binary object, Pattern Recognition 29 (3) (1996) 471–476.
[24] P.S.P. Wang, Y.Y. Zhang, A fast and flexible thinning algorithm, IEEE Trans. Comput. 38 (5) (1989) 741–745.
[25] P. Wunsch, A.F. Laine, Wavelet descriptors for multiresolution recognition of handprinted characters, Pattern Recognition 28 (8) (1995) 1237–1249.
[26] G.C.-H. Chuang, C.-C. Jay Kuo, Wavelet descriptors of planar curves: theory and applications, IEEE Trans. Image Process. 5 (1) (1996) 56–70.
[27] D. Hearn, M.P. Baker, Computer Graphics, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[28] A. Bem, Computer Graphics, Dept. of Informatics, Univ. of Athens, 1989.
[29] M. Misiti, Y. Misiti, G. Oppenheim, J.M. Poggi, Wavelet Tollbox for Use With Matlab, Math Works inc, Mar., 1996.
[30] A. Khotanzad, Y.H. Hong, Invariant image recognition by Zernike moments, IEEE Trans. PAMI 12 (5) (1990) 489–497.
[31] M. Shridhar, A. Badreldin, High accuracy character recognition algorithm using fourier and topological descriptors, Pattern Recognition 17 (5) (1984) 515–524.
[32] Timothy Masters, Practical Neural Networks, Recipes in C^{++}, Academic Press, New York, 1993.
[33] M.F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.
[34] V.G. Gezerlis, S. Theodoridis, A post-classification scheme for an OCR system for the notation of the orthodox hellenic Byzantine Music, Proceedings of Eusipco-2000 Conference, Finland, Sept. 2000.
[35] V.G. Gezerlis, Pattern recognition on the notation of the hellenic Byzantine Music, Master Thesis, Dept. of Informatics, University of Athens, 1996.

About the Author—VELISSARIOS G. GEZERLIS was born in Athens, Greece in 1971. He received a B.Sc. degree in Computer Science from the University of Athens, Department of Informatics and Telecommunications in 1993. In 1996 he received an M.Sc. degree in Advanced Information Systems and he is currently working for his Ph.D. in the area of Pattern Recognition, Optical Character Recognition and Byzantine Music Technology. He also holds the Diploma of Byzantine Music. He is currently working as a telecommunications engineer in the Hellenic Telecommunications Organization.

About the Author—SERGIOS THEODORIDIS received an honours degree in Physics from the University of Athens and his M.Sc. and Ph.D. degrees from the Department of Electronics and Electrical Eng. of Birmingham University, U.K. He is currently Professor of Signal Processing and Communications in the Department of Informatics and Telecommunications of Athens University. His Research interests lie in the areas of Adaptive Algorithms, Channel Equalization and Echo Cancellation, Pattern Recognition, Signal Processing for Music and OCR systems. He has published more than 100 papers in prestigious International journals and refereed

Conferences. He is the co-editor of the book “Efficient Algorithms for Signal Processing and System Identification”, Prentice-Hall 1993, the co-author of the book “Pattern Recognition”, Academic Press, 1998, and three books in Greek, two of them for the Greek Open University. He was the organizing committee chairman for the PARLE-94 Conference and the general chairman for EUSIPCO-98. He has served as an Associate Editor for the IEEE Transactions on Signal Processing and he is in the Editorial Boards of the Signal Processing and Applied Signal Processing journals. He was the Guest Editor for the special issue on Adaptive Algorithms and Echo Cancellation for the IEEE Signal Processing Magazine, he is a member of the ADCOM of the European Association for Signal and Image Processing (EURASIP) and he is a Fellow of the IEE.