OPTICAL CHARACTER RECOGNITION OF THE
ORTHODOX HELLENIC BYZANTINE MUSIC NOTATION

Velissarios G. Gezerlis and Sergios Theodoridis

University of Athens, Department of Informatics,
Division of Communication and Signal Processing,
Panepistimioupolis, TYPA Buildings, 157 84, Athens, Greece.
e-mail: gbelis@di.uoa.gr, stheodor@di.uoa.gr

Abstract: In this paper we present for the first time, the development of a new system for the off-line optical rec-
ognition 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 devel-
oped for the needs of the system.

Optical music recognition Off-line character recognition, Byzantine Music,
Byzantine Music Notation, Wavelets, Projections, Neural Networks
Contour processing, Nearest Neighbor Classifier Byzantine Music Data Base

1. INTRODUCTION

Byzantine Music is the type of music that was developed, formed and cultured, in the area of the Hellenic Ortho-
dox Christian Church, flourished mainly in the Byzantine epoch, continued its course in the afterwards me-
tabyzantine years and evolved, throughout the years, to the form which is known to day, serving always the war-
ship needs of the Orthodox Church.

* Author to whom correspondence should be addressed.
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 [32], [33].

A lot of research effort has been invested over the last forty years, for the development of systems that would be able to understand and recognize the western music notes. In the mid of 1960s, the first published work by Pruslin [27] 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 [28-30] 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 data base built. This data base 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 twenty projects have taken place and many hardware and software systems have been developed with the ability of music understanding, music recognition and music performance [31], [2].

The goal of this work is to develop a new optical recognition system for the Notation of the Hellenic Orthodox Byzantine Music. We will present an OBMR (Optical Byzantine Music Recognition) system that constitutes the first effort in the area of the Byzantine Music [37].

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 300dpi. 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 [12], [17], [9].
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 the third section we describe the structure of the OBMR system as well as the database of the bitmaps of the BMN. In section four, 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.

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 2,500 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 [32]. 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.
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 into 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 to be 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 $45^\circ$, $90^\circ$, $135^\circ$, or $180^\circ$ (e.g., the symbols «petasti» and «elafro» differ by a relative $180^\circ$ rotation).
- 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 [4]. 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 [21-22] 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 into 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 data base that works as a grammar and contains all the possible groups of the BMN (more than 2,500). 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.

### 3.1 Description of the Data Base 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 data base is about 71MB 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. 3.1 gives an example of different printed writing styles of the character «ison».

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. 3.2.

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 the printed ones and also to have some kind of variation. In fig. 3.3 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 3.4 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 [10], 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 [11].

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 in-
ternal 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. 4.1.1. 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 [15]. According to this method, when a 3x3 window matches the pattern of Fig. 4.1.2(a) and (b) the central pixel is filled (hole elimination). Filling is also carried out when the window matches the rotated pattern (b) in $90^\circ$, $180^\circ$, and $270^\circ$ degrees. On the other hand, deletion of the central pixel is carried out when a 3x3 window of the bitmap character matches the patterns (c) and (d) as well as the six equivalent patterns obtained from $90^\circ$, $180^\circ$ and $270^\circ$ degree 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 72x72. This is due to the following reason. If the size is greater than 72x72, 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 72x72, then the normalized characters are very small, so that some of their structural features may be lost. We developed a simple algorithm, which extracts the square bounding box of the character and scales the character to the 72x72 dimension. This makes the character images invariant to translations and scaling (see fig. 4.1.3) [1-4].

### 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 [5]. 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 [4]. There are also other methods for the $E_N$ computation, e.g., [3], [14].

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. 4.1.1.iv, 4.1.1.v). So, these classes correspond to more than one Euler numbers (see fig. 4.2.1). 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. 4.2.1, will participate into the first main subset, if its $E_N = 1$, and into the second main subset, if its $E_N = 0$. 
**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 [4]. 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 [3], [4].

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 [4].

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

\[
a x + b y + c = 0
\]

where

\[
a = C_y - P_y, \quad b = C_x - P_x
\]

and

\[
c = -(C_x - P_x) \cdot C_y + (C_y - P_y) \cdot C_x
\]

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 \(\vartheta\). 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 \(\vartheta\) is to use the 2nd order central moments \(\mu_{pq}\) from the equation [3], [5]:

\[9\]
\[ \vartheta = \frac{1}{2} \tan^{-1}\left( \frac{2\mu_{21}}{\mu_{20} - \mu_{02}} \right) \]

where central moments are given from:

\[ \mu_{pq} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i - C_j)^p (j - C_j)^q f(i, j) \]

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, \( \vartheta \), (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 preclassification schema:

If \( E_N \neq 1 \) (our character has one or two holes) then

1. If \( 20^0 \leq \vartheta \leq 72.9^0 \) then Slop = 1 (R-Diagonal)
2. If \( 72.9^0 < \vartheta \leq 118^0 \) then Slop = 2 (Vertical)
3. If \( 118^0 < \vartheta \leq 164.9^0 \) then Slop = 3 (L-Diagonal)
4. If \( 164.9^0 < \vartheta \leq 72.9^0 \) then Slop = 0 (Horizontal)
5. Else Slop = 4

Else If \( E_N = 1 \) (our character has no holes) then

1. If \( 10^0 < \vartheta \leq 80^0 \) then Slop = 1 (R-Diagonal)
2. If \( 80^0 < \vartheta \leq 121^0 \) then Slop = 2 (Vertical)
3. If \( 121^0 < \vartheta \leq 163.5^0 \) then Slop = 3 (L-Diagonal)
4. If \( 163.5^0 < \vartheta \leq 190^0 \) then Slop = 0 (Horizontal)
5. Else Slop = 4

This is a preclassification schema based on Euler number and direction of PA. We quantized the space \( 0^0 \) to \( 360^0 \) into 4 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 4 main slops. For characters that do not contain holes, we have found different values for the angles (including also
their supplementary angles). Figure 4.2.2 shows four characters with their PA and the 4 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 belong 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 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 [3]. If $a$ is the width of the horizontal bounding rectangle and $b$ the corresponding height, then we have the Ratio_of_HBR (see fig. 4.2.3). If this ratio is greater than 1, then the character is extended vertically (fig. 4.2.2.c). On the other hand, if the ratio is less than 1, then the character is extended horizontally (fig. 4.2.2.a) and if the ratio is approximately equal to 1, then the character is diagonal or square (fig. 4.2.2.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 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 2.1). 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 2.2). 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 2.3 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 $E_N = 1$. 

11
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 2.2, 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.

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. 4.2.1.2), we developed an algorithm based on the technique given in [18]. In this algorithm we used an adaptive starting point depending on the EN.

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 [8].

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. 4.2.1.1.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.
4.2.1.1.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.,

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

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 [7-8]. 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 [25], [26], 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 give also 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 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. 4.2.1.3.c.
Using the approximated version of the Bezier curves the resulting vectors have a dimension of 128 points, that is $x(i), i=0...127$ and $y(i), i=0...127$, 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 4 main directions. As it is pointed out in [1], the aim of the projection method is to simplify drastically a system of character recognition by reducing two-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 [12] 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. 4.2.1.4). 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 $\theta$.

The projection in a direction, e.g., vertical, is defined as the function $V_p(i), i=0...N-1$, 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 $H_p$ and the vertical projection $V_p$, 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. 4.2.1.1, used for the computation of the adaptive starting point. The $V_p$ and the $H_p$ are of dimension $N = 72$, while the $LDP$ and the $RDP$ have dimension $2N-1 = 143$ (fig. 4.2.1.5.a).

Using the adaptive projection method, we extract four vectors $P_{H}, P_{V}, P_{LDP}, P_{RDP}$, which correspond to the non-zero values of the horizontal, vertical, left and right diagonal projections respectively, as it is shown in fig. 4.2.1.5.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. 4.2.1.3.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 $P_{H(i)}, P_{V(i)}, P_{LDP(i)}, P_{RDP(i)}, i=0...127$. 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 \( x(i), i = 0...127 \) and constitutes the \( x \)-coordinate of the contour function, the second one is the vector \( y(i), i = 0...127 \), 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 \( i = 0...127 \).

For the discrete decomposition of the DWT we have used Daubechies «db2» filters, [34] with the following values for the low and high pass filters (see fig. 4.2.1.6):

We applied the Wavelet Decomposition onto the above six vectors, according to the algorithm given in [5], [7-8], [34]. The algorithm is as the following:

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 \( (0...127) \) we keep the values \( (0, 2, 4, 6, ..., 126) \), i.e., 64 points. We keep the 64 points from the high pass and repeat the same procedure for the other 64 low passed points. The above algorithm is repeated 7 times as one can see from fig. 4.2.1.7.

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 to 7 of the decomposition are used [7].

The same schema is followed for the decomposition of the \( y \) contour vector of length 128, as well as for the 4-projection vectors \( P_{H}(i), P_{V}(i), P_{LD}(i), P_{RD}(i) \), \( i = 0...127 \), 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 \( P_{16}(i) \) 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 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 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 \( F = \{ f_1, f_2, ..., f_m \} \), with the label of the nearest neighbor of \( F \) among all the training patterns [13]. 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 \( T^c = \{ t_{k_1}^c, t_{k_2}^c, ..., t_{km}^c \} \), where \( m = 48, k = 1, ..., 50, c = 1, ..., 71 \).

Let \( F \) be an unknown test feature vector. Then \( F \) is classified into the class \( C^* \) if:
\[ C' = \min_{c=1,..,C} d(F,T^c) \]

where \( d \) is the Euclidean Distance between the test vector \( F \) and the vector \( T^c \) equal to:

\[
d = \sqrt{(f_1 - t_{k1}^c)^2 + (f_2 - t_{k2}^c)^2 + \cdots + (f_m - t_{km}^c)^2}
\]

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 \( c \) [13]. 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 [20] (fig. 4.2.2.1). As we can see from figure 4.2.2.1, 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.

5. EXPERIMENTAL RESULTS

For our experiments we used the 18000 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 14200 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 2.1, 2.2 and 2.3 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 5.1 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 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 5.2). 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 5.3 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 Near-
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 difficult task. This set of classes is shown in Table 5.4. The average recognition accuracy for this set, according to Table 5.3 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. 5.1). For each neuron the log-sigmoid transfer function was used. For the training the Back-propagation algorithm was adopted [5], [35], in the form of the scaled conjugate gradient algorithm (SCG) presented in [36], 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., [5], [35], 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 5.5 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 5.5, 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.

A Post-Classification Scheme

A close observation of Table 5.3 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. 5.2): (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., [38]. 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. 5.2.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. 5.3, 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 character is the «fthora of Zo», otherwise the character is the «fthora of Vw».

For the confusion set of fig 5.2.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. 5.4. Then if the angle $\theta$ between the line segments $P_1P_{max}$ and $P_2P_{max}$ is greater than a predefined threshold $T_\theta$, then the character is the «fthora of Ni», otherwise it is the «fthora of Ni*».

For the character confusion set of fig. 5.2.c, we developed a different algorithm that estimates the line which 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. 5.5. We created a vector $C_l$ that contains the distances $d$ of the contour pixels of the left part, from the line $SpP$ and a vector $C_r$ 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. 5.2.c). If there are large values in the first 8 positions then we have the character «fthora of Di**» (the second one in fig. 5.2.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. 5.2.c). This algorithm exploits the symmetric form of the three characters as well as their minor differences.

For the character set of fig. 5.2.d and 5.2.e, we used the same algorithm since these characters are similar and differ only by a relative $180^\circ$ rotation. The algorithm follows the same philosophy of the previous described algorithm. The only difference is that in this algorithm we search the smoothed vector $C_d$ and $C_b$ and count the num-
ber 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 thresholds, we can separate the three characters of fig. 5.2.d and the ones of fig. 5.2.e.

Concerning the set of fig. 5.2.f, we used the same algorithm described for those of fig. 5.2.c. For the cases of 5.2.g and 5.2.h we make a second classification using only the contour information for the final feature vector. Finally, the separation of «apli» and «stigmi» (fig. 5.2.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 the figure 5.2, we obtain an average accuracy of 99.1% shown in table 5.6.

If we compare the two tables 5.3 and 5.6 in the positions of confused symbols we observe the large improvement in the accuracy.

**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 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 5.6. The second one was written very quickly with single strokes and large variation (cursive script), fig. 5.7. The results were 92.85% recognition for the fig. 5.6 and 73.2 % for the fig 5.7. 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 optical 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 nth 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 Multileyer 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.

REFERENCES


### TABLES

**Table 1:** The 71 distinct symbols of the Byzantine Music Notation.

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

* used for distinguishing the similar names

**Table 2.1:** Subsets when \(E_n = -1\) or \(-2\), and Slop = 1 2 3 0

<table>
<thead>
<tr>
<th>Slop</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

4 subsets with number of elements
Table 2.2: Subsets when EN = 0, and Slop = 0

<table>
<thead>
<tr>
<th>Slope</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset 1</td>
<td>91</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Subset 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Subset 3</td>
<td>91</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Subset 4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Subset 5</td>
<td>91</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 2.3: Subsets when EN = 1, and (Slop, Ratio of HBR) =

<table>
<thead>
<tr>
<th>Slope</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset 1</td>
<td>0-0.4</td>
<td>0.4-0.6</td>
<td>0.6-</td>
<td>0-0.8</td>
<td>0.8-1.6</td>
<td>1.6-</td>
<td>-</td>
<td>0-0.8</td>
<td>0.8-</td>
</tr>
<tr>
<td>Subset 2</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 3</td>
<td>99.5</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 4</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 5</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 6</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 7</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 8</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 9</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Subset 10</td>
<td>100</td>
<td>90.5</td>
<td>97.5</td>
<td>99.5</td>
<td>100</td>
<td>97.5</td>
<td>95.5</td>
<td>93.5</td>
<td>99.5</td>
</tr>
</tbody>
</table>

Table 5.1: The 71 symbols of table 1 and the respective recognition accuracies using only the contour information.

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

Average accuracy = 97.2 %
Table 5.2: 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%.

<table>
<thead>
<tr>
<th>N</th>
<th>Features for Character «trigrammi yfesi», «$P$»</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>16wt X + 16wt Y + 16wt $P_V$ + 16wt $P_H$ + 16wt $P_{1D}$ + 16wt $P_{RD}$</td>
<td>74.5%</td>
</tr>
<tr>
<td>48</td>
<td>16wt X + 16wt Y + 16wt $P_V$</td>
<td>69%</td>
</tr>
<tr>
<td>48</td>
<td>16wt X + 16wt Y + 16wt $P_H$</td>
<td>67%</td>
</tr>
<tr>
<td>48</td>
<td>16wt X + 16wt Y + 16wt $P_{1D}$</td>
<td>61%</td>
</tr>
<tr>
<td>48</td>
<td>16wt X + 16wt Y + 16wt $P_{RD}$</td>
<td>86%</td>
</tr>
<tr>
<td>64</td>
<td>16wtX + 16wtY + 16wt $P_V$ + 16wt $P_{RD}$</td>
<td>76.5%</td>
</tr>
<tr>
<td>32</td>
<td>16wt X + 16wt Y</td>
<td>78.5%</td>
</tr>
<tr>
<td>16</td>
<td>16wt $P_{RD}$</td>
<td>81%</td>
</tr>
</tbody>
</table>

Table 5.3: 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.

<table>
<thead>
<tr>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>19</td>
<td>97.5</td>
<td>37</td>
<td>98.5</td>
<td>55</td>
<td>96</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>20</td>
<td>100</td>
<td>38</td>
<td>86</td>
<td>56</td>
<td>99</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>21</td>
<td>100</td>
<td>39</td>
<td>98.5</td>
<td>57</td>
<td>92.5</td>
</tr>
<tr>
<td>4</td>
<td>99</td>
<td>22</td>
<td>100</td>
<td>40</td>
<td>96.5</td>
<td>58</td>
<td>97</td>
</tr>
<tr>
<td>5</td>
<td>97</td>
<td>23</td>
<td>99</td>
<td>41</td>
<td>98.5</td>
<td>59</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>24</td>
<td>99</td>
<td>42</td>
<td>99</td>
<td>60</td>
<td>98.5</td>
</tr>
<tr>
<td>7</td>
<td>99</td>
<td>25</td>
<td>100</td>
<td>43</td>
<td>100</td>
<td>61</td>
<td>98.5</td>
</tr>
<tr>
<td>8</td>
<td>99.5</td>
<td>26</td>
<td>100</td>
<td>44</td>
<td>99.5</td>
<td>62</td>
<td>99.5</td>
</tr>
<tr>
<td>9</td>
<td>99</td>
<td>27</td>
<td>100</td>
<td>45</td>
<td>99.5</td>
<td>63</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>99.5</td>
<td>28</td>
<td>96</td>
<td>46</td>
<td>100</td>
<td>64</td>
<td>98.5</td>
</tr>
<tr>
<td>11</td>
<td>92.5</td>
<td>29</td>
<td>99.5</td>
<td>47</td>
<td>99</td>
<td>65</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>30</td>
<td>98.5</td>
<td>48</td>
<td>96</td>
<td>66</td>
<td>99.5</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>31</td>
<td>95.5</td>
<td>49</td>
<td>92.5</td>
<td>67</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>96</td>
<td>32</td>
<td>99.5</td>
<td>50</td>
<td>94.5</td>
<td>68</td>
<td>99.5</td>
</tr>
<tr>
<td>15</td>
<td>99.5</td>
<td>33</td>
<td>98</td>
<td>51</td>
<td>100</td>
<td>69</td>
<td>99</td>
</tr>
<tr>
<td>16</td>
<td>99.5</td>
<td>34</td>
<td>100</td>
<td>52</td>
<td>97</td>
<td>70</td>
<td>96</td>
</tr>
<tr>
<td>17</td>
<td>96</td>
<td>35</td>
<td>100</td>
<td>53</td>
<td>94</td>
<td>71</td>
<td>96.5</td>
</tr>
<tr>
<td>18</td>
<td>99</td>
<td>36</td>
<td>99.5</td>
<td>54</td>
<td>86</td>
<td>72</td>
<td>96.5</td>
</tr>
</tbody>
</table>

Average accuracy = 98.1 %

Table 5.4: A set of 15 classes having many similarities.

Table 5.5: The network architecture input/1stHL/2ndHL/ output, the number of epochs where the network error is less than $10^{-5}$, and the network accuracy.

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Epochs</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 / 150 / 100 / 15</td>
<td>365</td>
<td>95.20</td>
</tr>
<tr>
<td>48 / 100 / 50 / 15</td>
<td>628</td>
<td>95.43</td>
</tr>
<tr>
<td>48 / 55 / 35 / 15</td>
<td>1069</td>
<td>94.53</td>
</tr>
<tr>
<td>48 / 47 / 30 / 15</td>
<td>1345</td>
<td>93.66</td>
</tr>
</tbody>
</table>

27
Table 5.6: Experiment employing the post classification schema. The final accuracy is 99.1%.

<table>
<thead>
<tr>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
<th>id</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>19</td>
<td>100</td>
<td>37</td>
<td>100</td>
<td>55</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>20</td>
<td>100</td>
<td>38</td>
<td>100</td>
<td>56</td>
<td>99</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>21</td>
<td>100</td>
<td>39</td>
<td>99.5</td>
<td>57</td>
<td>92.5</td>
</tr>
<tr>
<td>4</td>
<td>99</td>
<td>22</td>
<td>100</td>
<td>40</td>
<td>98</td>
<td>58</td>
<td>97</td>
</tr>
<tr>
<td>5</td>
<td>97</td>
<td>23</td>
<td>99</td>
<td>41</td>
<td>98.5</td>
<td>59</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>99</td>
<td>24</td>
<td>99</td>
<td>42</td>
<td>99</td>
<td>60</td>
<td>98.5</td>
</tr>
<tr>
<td>7</td>
<td>99</td>
<td>25</td>
<td>100</td>
<td>43</td>
<td>100</td>
<td>61</td>
<td>98.5</td>
</tr>
<tr>
<td>8</td>
<td>99.5</td>
<td>26</td>
<td>100</td>
<td>44</td>
<td>99.5</td>
<td>62</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>99</td>
<td>27</td>
<td>100</td>
<td>45</td>
<td>99.5</td>
<td>63</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>99.5</td>
<td>28</td>
<td>98.5</td>
<td>46</td>
<td>100</td>
<td>64</td>
<td>98.5</td>
</tr>
<tr>
<td>11</td>
<td>92.5</td>
<td>29</td>
<td>99.5</td>
<td>47</td>
<td>99</td>
<td>65</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>30</td>
<td>99</td>
<td>48</td>
<td>97</td>
<td>66</td>
<td>99.5</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>31</td>
<td>100</td>
<td>49</td>
<td>99</td>
<td>67</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>96</td>
<td>32</td>
<td>99.5</td>
<td>50</td>
<td>100</td>
<td>68</td>
<td>99.5</td>
</tr>
<tr>
<td>15</td>
<td>99.5</td>
<td>33</td>
<td>100</td>
<td>51</td>
<td>100</td>
<td>69</td>
<td>99</td>
</tr>
<tr>
<td>16</td>
<td>99.5</td>
<td>34</td>
<td>100</td>
<td>52</td>
<td>98.5</td>
<td>70</td>
<td>96</td>
</tr>
<tr>
<td>17</td>
<td>995</td>
<td>35</td>
<td>100</td>
<td>53</td>
<td>99</td>
<td>71</td>
<td>96.5</td>
</tr>
<tr>
<td>18</td>
<td>99.5</td>
<td>36</td>
<td>99.5</td>
<td>54</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average accuracy = 99.1%

FIGURES

Figure 1: A psalm of the Byzantine Music which is called «Christ is Risen». 
Figure 2: Twelve character groups of the BMN.

Figure 3: The structure of the OBMR system in abstract form.

Figure 3.1: Different variations of the printed character «ison».

Figure 3.2: Character samples of class «ison» written by the Byzwriter 1.0 software.

Figure 3.3: Character samples of class «ison» written carefully by hand.
Figure 3.4: Samples of the BMN database.

Figure 4.1.1: The five categories of noise, appeared in the character samples of the BMN.
Figure 4.1.2: The 3x3 patterns used for edge smoothing. Here «M» represents a black pixel. «o» represents a background pixel and «?» represents either

(a)                      (b)                      (c)                      (d)

Figure 4.1.3: a) The binarized character of dimension \( N_1 \times N_2 \). b) The normalized character of dimension 72x72.

(a)                      (b)

Figure 4.2.1: a) The character «delta» with \( E_N = 0 \). b) The character «delta» with \( E_N = 1 \), due to noise.

(a)                      (b)

Figure 4.2.2: 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 \( E_N = 1 \).

(a)                      (b)                      (c)                      (d)

Figure 4.2.2: Horizontal Bounding Rectangle, \( b = \) height, \( \alpha = \) width. If \( \text{Ratio of HBR} > 1 \) (c) then the character is extended vertically, if \( \text{Ratio of HBR} < 1 \) (a) is extended horizontally and if the \( \text{Ratio of HBR} \approx 1 \) (b) then the character is extended diagonally or is a square.

(a)                      (b)

Figure 4.2.1.1: Adaptive Starting Point. a) Left Diagonal Starting point (left diagonal scanning) if \( E_N = 1 \), b) Right Diagonal Starting point (right diagonal scanning) if \( E_N \neq 1 \).
Figure 4.2.1.2: The character «fhora of Pa*», its contour and the Right diagonal starting point.

Figure 4.2.1.3: 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 1st order with $128 = 2^7$ points.

Figure 4.2.1.4: 4 - Projections. The character «trigrami diesi» and its 4 - projections (we filled the internal of the hole with black color for the projection).

Figure 4.2.1.5: a) The 4-projections of «trigrammi diesi». The lines between the projections corresponds to the zero value. b) The four projection vectors $P_H$, $P_V$, $P_{LD}$, $P_{RD}$, without the zero values.

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

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

Figure 4.2.1.6: The Decomposition filter «db2» Daubechies 2. Low pass and High pass.
Figure 4.2.1.7: The discrete wavelet decomposition of vector \( x(i) \), where \( i = 0...127 \). The length of \( x \) is \( 128 = 2^7 \), that corresponds to 7 levels of decomposition.

Figure 4.2.2.1: 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 «yporo».

Figure 5.1: A Multilayer Perceptron with 2 hidden layers, 1 input with 48 nodes and 1 output with 15 nodes.
Figure 5.2: 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.

Figure 5.3: The algorithm that solves the confusion of fig. 5.2.a. The upper part of the two characters.

Figure 5.4: The algorithm that solves the confusion of fig. 5.2.b. The upper part of the two characters.

Figure 5.5: The algorithm that solves the confusion of fig. 5.2.c. The left part of the contour of character «zygos» and the line between the points $Sp$ and $P$. 
Figure 5.6: The psalm «Christ is Risen» in carefully handwritten form and recognition accuracy 92.85%.

Figure 5.7: The psalm «Christ is Risen» in cursive handwritten script and recognition accuracy 73.2%.