IJDAR (2003) 6: 10–22
Digital Object Identifier (DOI) 10.1007/s10032-003-0102-3

International Journal on IJDAR Document Analysis and Recognition

A new graph-like classification method applied to ancient handwritten musical symbols

João Caldas Pinto, Pedro Vieira, João M. Sousa

Technical University of Lisbon, Instituto Superior Técnico, IDMEC/IST, Av. Rovisco Pais, 1049-001 Lisbon, Portugal

Received: 27 December 2002 / Accepted: 10 January 2003
Published online: 4 July 2003 – © Springer-Verlag 2003

Abstract. Several algorithms have been proposed in the past to solve the problem of binary pattern recognition. The problem of finding features that clearly distinguish two or more different patterns is a key issue in the design of such algorithms. In this paper, a graph-like recognition process is proposed that combines a number of different classifiers to simplify the type of features and classifiers used in each classification step. The graph-like classification method is applied to ancient music optical recogniti on, and a high degree of accuracy has been achieved.

Keywords: Binary image segmentation – Binary pattern recognition – Feature selection – Optical music recognition

1 Introduction

Optical music recognition (OMR) is the process of identifying music from an image of a music score [2,4]. OMR has many similarities with optical character recognition (OCR) [18]. This recognition technique finds out the characters in the image, while OMR tries to identify the musical symbol in the image including notes, rests, clefs, accidents, etc. When compared to classical OMR systems, handwritten music recognition introduces additional difficulties in recognizing music symbols. The most important are the following: (1) notation varies from writer to writer; (2) simple (but important) and large changes in notation can occur in the same score; (3) staff lines mostly do not have the same height and are not always straight; (4) symbols are written with different sizes, shapes, and intensities; (5) the relative size between different components of a musical symbol can vary; (6) more symbols are superimposed in handwritten music than in printed music; (7) different symbols can appear connected to each other, and the same musical symbol can appear in separated components; and (8) paper degradation requires specialized image-cleaning algorithms. Due to these difficulties only some attempts to tackle this problem have been found in the literature, and works by Leplumey et al. [11] and Pèpin et al. [14] are illustrative examples. However, the general OMR problem for printed music has been dealt with by different authors; see, for example, [1,12,15].

This paper proposes an OMR method developed under the Portuguese project ROMA (Ancient Music Optical Recognition), which intends to recover manuscripts of ancient music scores to obtain a digital, easy-to-manage, easy-to-conserve, and, last but not least, easy-to-handle music heritage. The proposed system recognizes ancient handwritten music scores to obtain a digital, easy-to-manage version of ancient music. The methodology proposed in this paper is a staged process consisting of the following steps: (1) preprocessing of the image, (2) segmentation, (3) recognition, and (4) reconstruction. The first stage prepares the image for segmentation by cleaning and correcting some defaults. The segmentation divides the image of the music sheet in zones of interest, such as staff lines and bars, and isolates each musical symbol. Then, the recognition classifies each musical symbol. Finally, a process of reconstruction is needed to relate the recognized symbols with each other and with its staff lines and bars, creating the final description of the music.

The paper is organized as follows. Section 2 presents the segmentation used in this paper. Section 3 presents the recognition approach proposed in this paper, which is divided into feature selection and classification. An application example using music sheets from the 16th century is described in Sect. 4. Finally, Sect. 5 concludes the paper and presents guidelines for future research.

2 Segmentation

The segmentation process is the first step in OCR. This paper extends a method developed recently for ancient music recovery [5]. Although applied to music data, this

Correspondence to: J.C. Pinto
(e-mail: jcpinto@dem.ist.utl.pt)

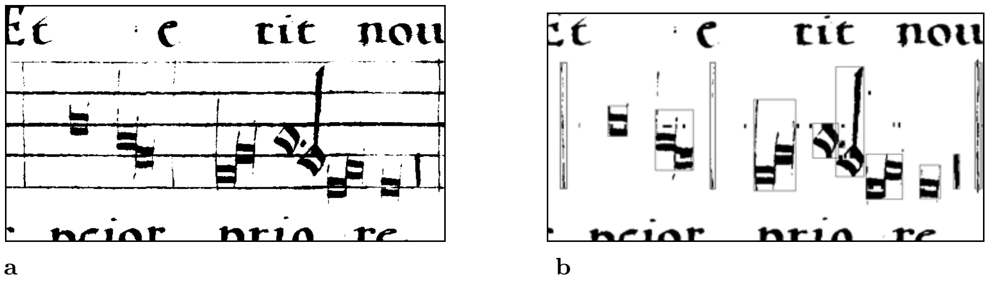


Fig. 1. Binary image of a piece of a musical sheet from the 16th century. **a** Before segmentation. **b** After segmentation

algorithm can be generalized to other types of data. The five-step segmentation method used in this paper is exemplified in Fig. 1, which contains an original binary image of a piece of a musical sheet from the 16th century.

The segmentation proposed by Pinto et al. [5] has the following steps.

1. **Identification and removal of staff lines**
   (a) *Identification of staff lines*: by using horizontal projections and small rotations of the image, this step finds the areas with peaks of the projections and classifies them as *staff lines.*
   (b) *Removal of staff lines*: knowing the location of the lines, the line width is estimated, and the segments of line whose thickness is not bigger than a certain threshold are removed. This threshold is proportional to the estimated line width and changes horizontally along the image since the line thickness also changes from left to right. Note that the removal of staff lines does not damage the symbols in Fig. 1b.
2. **Localization of bar lines.** Bar lines are defined as straight lines from the top to the bottom of the staff line. Using the vertical projection in each staff line, the peaks that correspond to bar lines are determined. These peaks are achieved after the application of a low filter to the vertical projection. Then each region corresponding to a peak is analyzed using its horizontal projection. Only regions with a high standard deviation of the horizontal projection are classified as bar lines.
3. **Object segmentation.**
   (a) *Labeling*: each group of six connected black pixels inside each staff line is identified and labeled.
   (b) *Expansion*: this process completes each object that crosses the frontier of the staff line, which contains portions outside the staff line.
4. **Removal of undesirable objects.** Objects that are too small, too big, or that intersect a text line are removed. This last type of object uses *text line search*, again using horizontal projection between staff lines. Areas with local maxima are identified and considered as text entering a staff line.
5. **Object merge.** This step uses domain knowledge.
   (a) *Vertical dashes*: this step uses morphological closure with a structuring element consisting of a vertical line (vertical closure) and joins vertical dashes with each other and with their objects. This method is necessary to merge objects that are separated by ink, scan, or binarization problems.
   (b) *Breve merge*: some objects in the data consist of connections of the note "breve" (see Fig. 2). Sometimes several breves are separated as vertical dashes. In order to merge them without merging undesirable objects, it is necessary to perform a severe vertical open [16] to retain only the vertical dashes of the breves, followed by a vertical closure to connect those dashes. The dashes that are connected correspond to objects that should be merged.
   (c) *Semibreve merge*: the "semibreve" note, presented in Fig. 3, is sometimes separated into two parts. This separation is due to the six-connectivity of the labeling process or simply to ink, scan, or binarization problems. A process based on the way the two bounding boxes intersect each other and on the difference between the areas of each object is used to classify the two objects as parts of the same semibreve.

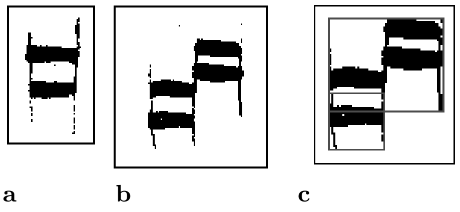


Fig. 2. **a** A breve note. **b** Two breve notes connected. **c** Two breve notes connected and separated by the labeling process

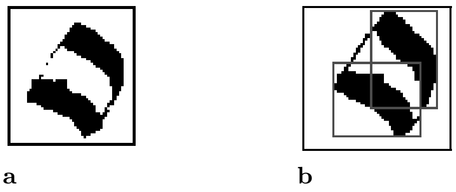


Fig. 3. **a** A semibreve note. **b** A semibreve separated by the labeling process

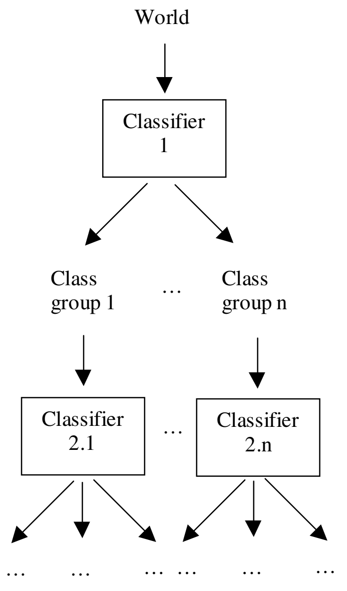


Fig. 4. Graph-like classification

3 Recognition

This paper proposes a new recognition process for OMR based on a graph structure of classifiers, as depicted in Sect. 3.1. The classifiers in the graph can be any type of classifier, such as a simple Bayes classifier [3], neural network classifiers [9], syntactic classifiers [7], or fuzzy classifiers [8].

Each classifier is divided into two steps:

1. *Feature extraction* determines the relevant features of the object.
2. *Classification* uses one of the classification methods described above and estimates a class for the object given the calculated features.

Note that this recognition method can be applied to any OCR, but it was especially developed for OMR in this paper. This method has some important characteristics:

1. Class hierarchy: the recognition process selects an object and follows the tree from the top to the bottom of the hierarchy. This process specializes the class of the object as it passes through the several classifiers. The leafs of the hierarchy are the final classes of the overall classification algorithm. This class hierarchy can be obtained directly from the graph of the recognizer by simply removing the classifiers' boxes in Fig. 4, as the graph structure of the recognizer and the class hierarchy are strictly related.
2. Set of classifiers: where each class that is not a leaf has one classifier, which classifies each object of that class into one of its subclasses.
3. Set of features: each classifier is based on a set of features, and it is important to study the optimum set, minimal and most accurate, that leads to acceptable results.

These issues are all related in the sense that each non-leaf class in the hierarchy represents a decision point of the classification process, and each decision is made by a classifier based on a set of features. Therefore, each class also has an associated feature map. This means that the design of a class hierarchy must take into account the choice of a set of features for each class as well as the classification algorithm, i.e., the three issues must be designed at the same time.

3.1 Design of a class hierarchy

The design of a class hierarchy consists of both maximizing the accuracy of the classification process and minimizing the complexity of the overall algorithm. The complexity is originated from two different sources: *the complexity of the graph*, which is a measure of the number of classes on the graph and the depth (number of steps) of the decision process, and *the complexity of each classifier*, which is the complexity of the classifier algorithm itself. This complexity comes from the intuitiveness of the algorithm, its training needs, and the complexity of the features associated with it, which is mainly the complexity of the algorithm that computes these features. Note that the less complex a graph is the more complex its classifiers must be in order to achieve the same classification accuracy.

Another issue concerning the design of the graph is the number of different paths from the top of the hierarchy to a given leaf. In a hierarchy where for each class there is only one path from the top of the hierarchy to that class, its structure is a tree instead of a graph. This means that two classes cannot share the same descendants in the hierarchy. This might seem to be the ideal situation since the total number of paths in the graph (in this case a tree) is not larger than the number of final classes (leafs). However, one can reduce complexity and raise intuitiveness by allowing some classes to share descendants, as we will see in Sect. 3.1.1.

The design of a class hierarchy can be defined as the grouping of leaf classes that have something in common (mostly feature values) in superclasses. The simplest hierarchy is the one with only the topmost class having the leaf classes as subclasses. In this case, the topmost class is associated with the single classifier of the whole graph, which must be extremely powerful in order to achieve a high level of accuracy. By grouping similar leaf classes in superclasses, a simpler classifier can be defined.

The classes must be grouped by similar feature values. Moreover, the feature values of each superclass must be as different as possible so that the classifier distinguishing among the superclasses can be simple. Given a feature set, clustering algorithms can be used to find the best groupings [9].

In summary, the design of a class hierarchy includes the set of classifiers and features for each class. The designer must group the classes and test them until good classes are found. This search for a hierarchy consists of the following steps:

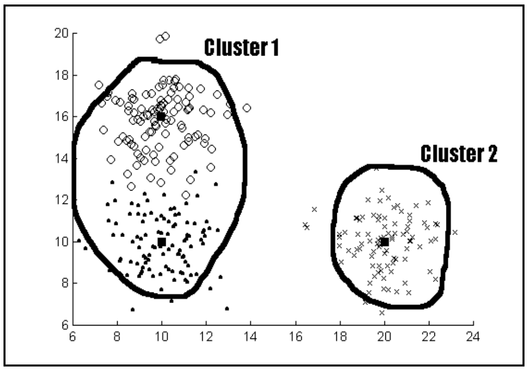


Fig. 5. Clusters obtained using the cluster by mean method

Repeat
(1) Group classes in superclasses using clustering algorithms.
(2) Measure the quality of the superclasses by evaluating the clusters obtained in (1).
(3) Test classifiers for the superclasses **until** the performance of the class hierarchy is accepted.

The description of these steps is presented in the next three sections.

3.1.1 Clustering classes. In this step, sets of classes with similar feature values are defined using clustering algorithms. A class is defined as a collection of objects (musical objects in this paper) and can be clustered by mean and by instances.

In *clustering by mean*, first the mean of each feature for each class is computed using a set of objects in the training set belonging to that class. Thus, each class is represented by its respective mean and constitutes one instance for the clustering algorithm. The clustering algorithm produces clusters of points in the feature space, where each point corresponds to one class. The clusters represent superclasses, and the cluster points represent subclasses of each superclass. Figure 5 represents three classes in a feature space with two hypothetical features. Each class has a mean in the feature space, represented by a black square. Two clusters are created considering only three points: one point per class.

In *clustering by instances*, each object of a class constitutes an instance of the clustering algorithm. The clustering algorithm produces clusters of points in the feature space, where each point corresponds to one object. One cluster may include objects from different classes, and two objects of the same class can belong to different clusters. This situation is presented in Fig. 6, where objects belonging to two classes are distributed by three clusters. Clustering by instances does not induce the set of superclasses and subclasses directly as in clustering by mean. This paper introduces the following two methods of extracting a class hierarchy from the results of the algorithm:

1. *MAX extraction.* First, one superclass is defined per cluster. Each original class is a subclass of one and only one superclass, which is the one where the cluster has more objects.
2. *COUNT extraction.* Again, one superclass is defined per cluster. When the number of objects inside a superclass cluster is above a certain threshold, called an *instance threshold*, each original class is a subclass of a superclass. If the instance threshold is equal to zero, every class that has at least one object in a superclass cluster is a subclass of that superclass.

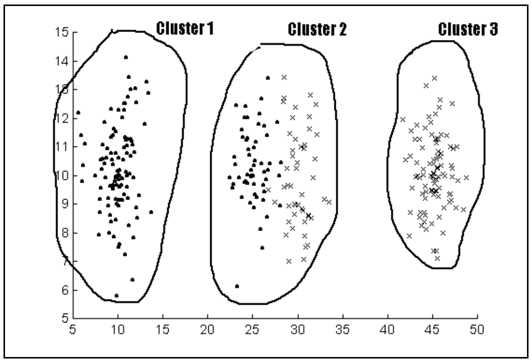


Fig. 6. Clustering using the *cluster by instances* method

Some important observations must be made:

1. Remember that a superclass always has one cluster associated with it. This means that a superclass contains a set of objects from the training set. However, while in cluster by mean a superclass contains all the objects of its subclasses, in cluster by instances a superclass contains only the objects of its cluster, which may not be the total set of objects of its subclasses. Recall that in cluster by instances each subclass may be distributed by several superclasses. This point is very important when creating classifiers to distinguish between superclasses (see Sect. 3.1.3). In Fig. 6, the classifier distinguishes between clusters 1, 2, and 3 but not class A from class B. Instead, it distinguishes between part of class A, part of class B, and two other parts of classes A and B together. This means that this method induces classifiers that distinguish different parts inside one class.
2. Cluster by means induces a simpler tree-like hierarchy and is normally used first. It assumes that each class has its objects distributed closely to a center (mean) in the chosen feature space. If it does not produce good clusters (see Sects. 3.1.2 and 3.1.3), either the chosen feature set should be changed or a more sophisticated method of clustering by instances should be chosen.
3. Cluster by instances is mostly used in cases where the objects of one or more classes are distributed around several centers. When more clusters (superclasses) than (sub)classes are defined, this method is preferable to clustering by mean. In Fig. 6, for instance, clusters 1 and 3 can immediately classify an object.
4. Clustering by instances can also be used to test clustering by mean; see Sect. 3.1.2.

Several clustering algorithms can be used such as k-means clustering [3], isodata [14], linear vector quantization [3], neural networks [9], or fuzzy clustering [8]. These methods produce a set of clusters that contain similar objects. The similarity is measured using the distance between two objects in the feature space. Each feature induces its distance measure, depending on a numeric value or a discrete value. The combination of the different feature distances can be made by several methods such as the Manhattan or euclidean distances [6].

3.1.2 Measuring clusters. When a set of superclasses is defined from a set of classes, the performance of this set must be measured. Then the quality of the clusters created is also being measured. In general, the best cluster set is the one that induces the simplest classifier with maximum accuracy. Several quality measures are presented below.

1. *Mean intraset distance* – the mean distance between two objects of the same cluster in the feature space [2]. The lower the intraset distance, the better the cluster set, i.e., clusters are better when the feature values of their objects are very similar. The intraset distance for a given cluster c is defined as:

$$D^c_{\text{intra}} = \frac{1}{M(M-1)} \times \sum_i \sum_{j \neq i} d^2(x_i, x_j)$$

where $d^2(x_i, x_j)$ is the distance between two objects in the feature space and M is the number of objects in cluster c. The mean intraset distance is given by:

$$\overline{D_{\text{intra}}} = \frac{1}{N_c} \times \sum_c D^c_{\text{intra}}$$

where N_c is the number of clusters in the cluster set.

2. *Mean interset distance* – the mean distance between two objects from different clusters in the feature space [2]. The bigger the interset distance, the better the cluster set, i.e., clusters are better when they are well separated in the feature space. The interset distance for two given clusters c_1 and c_2 is defined as:

$$D^{c_1,c_2}_{\text{inter}} = \frac{1}{M_1 M_2} \times \sum_i^{M_1} \sum_j^{M_2} d^2(x_i, x_j)$$

where M_1 and M_2 are the number of objects in clusters c_1 and c_2, respectively. The mean interset distance is given by:

$$\overline{D_{\text{inter}}} = \frac{1}{N_c(N_c-1)} \times \sum_{c_1} \sum_{c_2 \neq c_1} D^{c_1,c_2}_{\text{inter}}$$

3. *Intraset/interset distance* – a combination of both quality measures and defined as [3]:

$$D_{\text{intra/inter}} = \frac{\sum_c D^c_{\text{intra}}}{\sum_{c_1} \sum_{c_2 \neq c_1} D^{c_1,c_2}_{\text{inter}}}$$

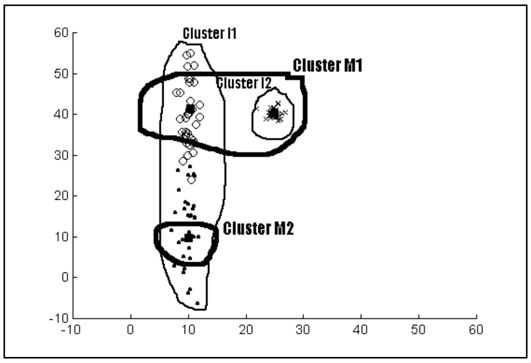


Fig. 7. Clustering by means (clusters M1 and M2) and by instances (clusters I1 and I2). Example of how clustering by instances separates the clusters created by the clustering by mean

The lower the intraset/interest measure, the better the cluster set.

Given these three quality measures one can compare different cluster sets in order to choose the one that best aggregates the classes into superclasses. The cluster sets can be easily compared when only the number of clusters varies. However, to compare cluster sets with different feature spaces, the distance between two objects must be normalized, which can be done by:

1. *Feature distance normalization.* When one feature is substituted by another, the feature distance measure must be normalized in order to compare the distances of two different features.
2. *Combinatory distance normalization.* When the number of features changes, the combinatory distance measure must be normalized. Note that simple Manhattan or euclidean distances cannot be used since they grow with the dimension of the feature space.

Figure 8 in Sect. 4.1 shows an example of a comparison table for several cluster sets, varying only the number of clusters.

The quality of a cluster set obtained using the cluster by mean method can be measured by using the cluster by instances method. This last method is applied to the same number of clusters obtained using the cluster by means. For instance, if eight classes are clustered into three superclasses using clustering by mean, then three superclasses can be clustered into three clusters by using clustering by instances. If these three clusters correspond to the three superclasses (one cluster per superclass only), then this is a good cluster set. If the three superclasses are distributed by more than one of the three clusters, then there is some mixture between classes in the cluster set and the cluster set is not acceptable. In Fig. 5, for instance, a clustering by instances would maintain the clusters created using the clustering by mean method. However, in Fig. 7, the clustering by instances separates the clusters obtained using the clustering by mean method.

3.1.3 Testing classifiers. When a cluster set is accepted as a good one by the quality measures defined in Sect. 3.1.2, it is necessary to derive a classifier that attributes one of the superclasses defined by the clusters to an object. Two simple classifiers are:

1. *Nearest prototype classifier*: this classifier is computed using the mean of the objects defined in the feature space of each cluster [6]. An object is classified in the superclass that minimizes the distance of its mean to the object.
2. *MAP (maximum a posteriori) classifier*: this classifier is computed using the mean μ and standard deviation σ of the objects in the feature space of each cluster [6]. An object is classified in the superclass where the probability of the object belonging to that class is greater than its belonging to any other class:

$$\widehat{\omega} = \arg\max_{\omega} P\left(\omega|x\right) \cdot P\left(\omega\right)$$

where $P\left(\omega\right)$ is the probability of a superclass being related to the other superclasses, which are estimated by the number of objects in the cluster over the total number of objects in the training set; $P\left(\omega|x\right)$ is the probability of having the class ω when x is given with:

$$P\left(\omega|x\right) \sim N\left(\mu, \sigma\right),$$

Other classifiers using, e.g., neural networks or fuzzy classifiers can be obtained. However, already having a feature set that divides the superclasses fairly well, it is sufficient to use very simple classifiers, such as the two presented, to achieve good results.

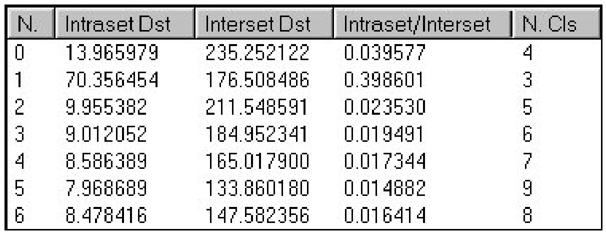
In order to test the quality of the classifier (and, therefore, the cluster set), a testing set of objects must be used. The number of objects classified in each class is counted, and confusion matrices are produced. The lines of these matrices correspond to the input classes (classes that the objects belong to) and the columns correspond to the output classes (classes in which the objects are classified). The value of line i and column j corresponds to the number or percentage of objects from input class i that were classified in output class j. The classifiers being tested distinguish superclasses, and thus the output classes are also always superclasses. Two types of confusion matrices have been used:

1. Standard confusion matrix: where the input and the output classes are the same, and the superclasses are created by the clustering process [10]. This matrix presents the percentage of well-classified and misclassified objects.
2. Detailed confusion matrix: the input classes of this matrix are the subclasses of the superclasses created, which gives the percentage of well-classified and misclassified objects discriminated by each subclass. With this matrix it is possible to know which subclasses are fully well classified and which are separated into two or more superclasses. In this case, the class hierarchy is changed by adding subclasses to the superclasses that were misclassified. Suppose that a superclass A has subclasses A1 and A2, and superclass B has subclasses B1 and B2. Furthermore, a certain classifier sometimes classifies class B1 in superclass A. Then the class hierarchy should be changed such that superclass A is divided into the subclasses A1, A2, and B1, and superclass B is divided into subclasses B1 and B2. Note that this procedure should only be done when the same classifier is used in the final recognition graph.

3.1.4 Automating the methodology. The methodology presented in Sects. 3.1.1, 3.1.2, and 3.1.3 was developed manually for the OMR application in this paper. However, it is possible to automate this process, as proposed in the following algorithm, which can be applied to OCR problems in general.

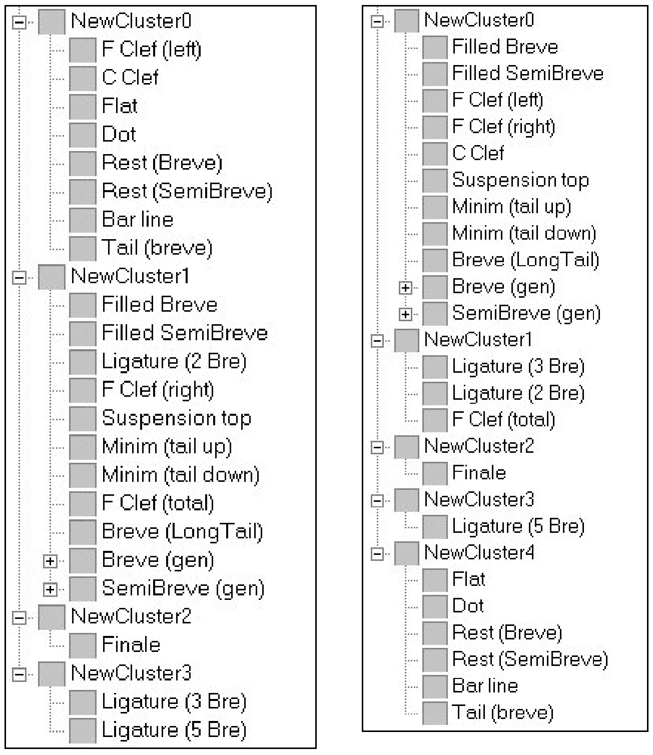
Algorithm: design of a class hierarchy
Input: classes, features, classifier types, training set, test set
Output: class hierarchy (superclasses, classifiers, and selected features)
Steps:

1. For each subset of *features*, cluster the *classes* into superclasses in the feature space for different numbers of superclasses by using the *training set.*
2. Choose the cluster sets that evaluate above a certain threshold using the quality measures and the normalizing distance measures in the *test set.*
3. Induce classifiers for the *classifier types* using the *training set.*
4. Compute the confusion matrices for each cluster set and for each induced classifier, i.e., test the class hierarchies using the *test set.*
5. Choose the "best" class hierarchy, considering the following measures:
   (a) The number of misclassifications; fewer misclassifications is preferred.
   (b) The number of induced copies of subclasses in superclasses: a subclass *sub1* from superclass *sup1* is induced to be copied to a superclass *sup2* (to become a subclass of *sup2*) if the percentage of misclassifications of *sub1* in *sup2* (in comparison with the number of correct classifications) exceeds a certain threshold; fewer induced copies is preferred.
   (c) The simplicity of the classifier (for instance, k-nearest neighbors) is computationally less efficient than the nearest prototype classifier; simpler classifiers are preferred.
   (d) The size and computational complexity of the feature set; fewer and simpler (more efficient) features are preferred.
6. Change the hierarchy by including the induced copies of subclasses into superclasses.



N.	Intraset Dst	Interset Dst	Intraset/Interset	N. Cls
0	13.965979	235.252122	0.039577	4
1	70.356454	176.508486	0.398601	3
2	9.955382	211.548591	0.023530	5
3	9.012052	184.952341	0.019491	6
4	8.586389	165.017900	0.017344	7
5	7.968689	133.860180	0.014882	9
6	8.478416	147.582356	0.016414	8

Fig. 8. Comparison of quality measures for different cluster sets



```
a
⊟ NewCluster0
    F Clef (left)
    C Clef
    Flat
    Dot
    Rest (Breve)
    Rest (SemiBreve)
    Bar line
    Tail (breve)
⊟ NewCluster1
    Filled Breve
    Filled SemiBreve
    Ligature (2 Bre)
    F Clef (right)
    Suspension top
    Minim (tail up)
    Minim (tail down)
    F Clef (total)
    Breve (LongTail)
  ⊞ Breve (gen)
  ⊞ SemiBreve (gen)
⊟ NewCluster2
    Finale
⊟ NewCluster3
    Ligature (3 Bre)
    Ligature (5 Bre)
```

```
b
⊟ NewCluster0
    Filled Breve
    Filled SemiBreve
    F Clef (left)
    F Clef (right)
    C Clef
    Suspension top
    Minim (tail up)
    Minim (tail down)
    Breve (LongTail)
  ⊞ Breve (gen)
  ⊞ SemiBreve (gen)
⊟ NewCluster1
    Ligature (3 Bre)
    Ligature (2 Bre)
    F Clef (total)
⊟ NewCluster2
    Finale
⊟ NewCluster3
    Ligature (5 Bre)
⊟ NewCluster4
    Flat
    Dot
    Rest (Breve)
    Rest (SemiBreve)
    Bar line
    Tail (breve)
```

a **b**

Fig. 9. Two cluster sets: **a** four clusters and **b** five clusters

4 Case studied

The design of a class hierarchy proposed in this paper was applied to an OMR problem, namely, a set of handwritten musical symbols, for a particular ancient notation from the 16th century, which is part of the book MM37 – *Canto Chão* from the Library of Coimbra University. The class hierarchy divides the set of handwritten musical symbols into several subclasses. It will be seen that simple features and simple decision rules can solve the problem completely. For illustration purposes two steps of the algorithm are presented. The first corresponds to the first clustering operation based on the bounding box width. The second step presents the feature evaluation procedure for clustering a particular subclass obtained in the first step.

The musical notation under study has 27 classes corresponding to musical objects including, e.g., figures, clefs, and rests. Some important objects were shown in Figs. 1, 2, and 3. The classes and number of objects are presented in Table 1.

Table 1. Classes and number of objects of the OMR problem in the training set

Class	Number of objects
Semibreve (gen)	1039
Breve (gen)	329
F Clef (right)	101
F clef (left)	93
Ligature (2 bre)	55
Rest (breve)	50
Minim (tail up)	29
Minim (tail down)	26
Flat	23
Dot	23
C Clef	21
Bar line	19
Tail (breve)	18
Rest (semibreve)	17
Ligature (3 bre)	10
F clef (total)	10
Filled breve	9
Filled semibreve	4
Suspension top	3
Ligature (5 bre)	2
Finale	1
Breve (long tail)	1
Quavers (2)	0
Sharp	0
Crotchet (tail up)	0
Crotchet (tail down)	0
Ligature (5 bre)	0
Total:	**1883**

Table 2. Standard confusion matrices using four clusters

Nearest prototype classifier					Maximum a posteriori classifier				
	Cl0	Cl1	Cl2	Cl3		Cl0	Cl1	Cl2	Cl3
Cl0	250	12	0	2	Cl0	254	8	0	2
Cl1	65	1529	0	12	Cl1	142	1390	0	74
Cl2	0	0	1	0	Cl2	0	0	1	0
Cl3	0	0	0	12	Cl3	0	0	0	12
No. of misclassifications: 91					No. of misclassifications: 126				

4.1 Clustering the overall set of musical objects

The k-means clustering algorithm and a feature set with only the bounding box width feature are used to divide the set of classes, considering that the simplest class hierarchy divides the simplest features at the top and the most complicated at the bottom. We tested from three up to nine clusters. The results for the quality measures presented in Sect. 3.1.2 are shown in Fig. 8.

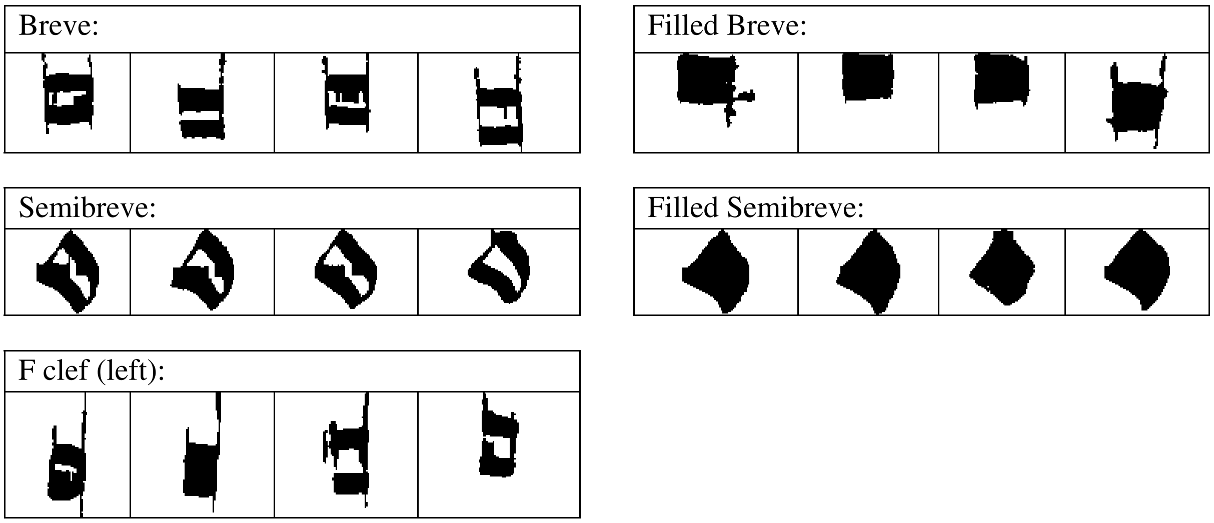


Fig. 10. Original images of musical symbols from the class BB_NS_Box

Table 3. Detailed confusion matrices using four clusters

	Nearest prototype classifier				Maximum a posteriori classifier			
	Cl0	Cl1	Cl2	Cl3	Cl0	Cl1	Cl2	Cl3
Cl0	↓			↓				
F clef (left)	81	11	0	1	85	7	0	1
C clef	20	1	0	0	20	1	0	0
Flat	23	0	0	0	23	0	0	0
Dot	23	0	0	0	23	0	0	0
Rest (breve)	50	0	0	0	50	0	0	0
Rest (semibreve)	17	0	0	0	17	0	0	0
Bar line	18	0	0	1	18	0	0	1
Tail (breve)	18	0	0	0	18	0	0	0
Cl1		↓			↓			
Filled breve	3	6	0	0	5	4	0	0
Filled semibreve	0	4	0	0	0	4	0	0
Ligature (2 bre)	0	48	0	7	0	7	0	48
F clef (right)	0	101	0	0	0	101	0	0
Suspension top	0	3	0	0	0	2	0	1
Minim (tail up)	0	29	0	0	1	28	0	0
Minim (tail down)	0	26	0	0	0	26	0	0
F clef (total)	0	5	0	5	0	0	0	10
Breve (longtail)	0	1	0	0	0	1	0	0
Breve (gen)	59	270	0	0	125	203	0	1
Semibreve (gen)	3	1036	0	0	11	1014	0	14
Cl2			↓				↓	
Finale	0	0	1	0	0	0	1	0
Cl3				↓				↓
Ligature (3 bre)	0	0	0	10	0	0	0	10
Ligature (5 bre)	0	0	0	2	0	0	0	2

Table 4. Standard confusion matrices using five clusters

Nearest prototype classifier						Maximum a posteriori classifier					
	Cl0	Cl1	Cl2	Cl3	Cl4		Cl0	Cl1	Cl2	Cl3	Cl4
Cl0	1572	19	0	1	63	Cl0	1558	33	0	1	63
Cl1	5	70	0	0	0	Cl1	0	75	0	0	0
Cl2	0	0	1	0	0	Cl2	0	0	1	0	0
Cl3	0	0	0	2	0	Cl3	0	0	0	2	0
Cl4	0	1	0	0	149	Cl4	0	1	0	0	149
No. of misclassifications: 89						No. of misclassifications: 98					

Table 5. Detailed confusion matrices using five clusters

	Nearest prototype classifier					Maximum a posteriori classifier				
Cl0	Cl0	Cl1	Cl2	Cl3	Cl4	Cl0	Cl1	Cl2	Cl3	Cl4
Cl0	↓				↓					
Filled breve	9	0	0	0	0	9	0	0	0	0
Filled semibreve	4	0	0	0	0	4	0	0	0	0
F clef (left)	40	0	0	1	52	40	0	0	1	52
F clef (right)	101	0	0	0	0	99	2	0	0	0
C clef	11	0	0	0	10	11	0	0	0	10
Suspension top	2	1	0	0	0	2	1	0	0	0
Minim (tail up)	29	0	0	0	0	29	0	0	0	0
Minim (tail down)	26	0	0	0	0	26	0	0	0	0
Breve (longtail)	1	0	0	0	0	1	0	0	0	0
Breve (gen)	327	1	0	0	1	326	2	0	0	1
Semibreve (gen)	1022	17	0	0	0	1011	28	0	0	0
Cl1	1022	↓					↓			
Ligature (3 bre)	0	10	0	0	0	0	10	0	0	0
Ligature (2 bre)	5	50	0	0	0	0	55	0	0	0
F clef (total)	0	10	0	0	0	0	10	0	0	0
Cl2			↓					↓		
Finale	0	0	1	0	0	0	0	1	0	0
Cl3				↓					↓	
Ligature (5 Bre)	0	0	0	2	0	0	0	0	2	0
Cl4					↓					↓
Flat	0	0	0	0	23	0	0	0	0	23
Dot	0	0	0	0	23	0	0	0	0	23
Rest (breve)	0	0	0	0	50	0	0	0	0	50
Rest (semibreve)	0	0	0	0	17	0	0	0	0	17
Bar line	0	1	0	0	18	0	1	0	0	18
Tail (breve)	0	0	0	0	18	0	0	0	0	18

Table 6. Quality measures for the two clusters created

Quality measure	Manhattan	Euclidean
Intraset distance	7.00	5.59
Interset distance	30.45	23.46
Intraset/interset	0.46	0.48

The best relation between intraset and interset distances is obtained using four and five clusters. The hierarchies obtained using these numbers of clusters is presented in Fig. 9.

Then the nearest prototype and MAP classifiers were tested. The confusion matrices using these methods are presented in Tables 2–5.

By analyzing the results in the confusion matrices, the hierarchy with five clusters using the MAP classifier is chosen based on the following:

1. The number of wrong classifications is inferior in the five-cluster hierarchy.
2. The MAP classifier obtained a classification accuracy of 100% in cluster 1. In cluster 0, both MAP and NP classifiers diminish their accuracy, due mostly to misclassifications of semibreves.

This problem will be solved changing the class hierarchy. Note that semibreves were often misclassified in cluster 1 and F clefs (left), and C clefs were often misclassified in cluster 4. Considering these misclassifications, the subclass *semibreve* has been added to the superclass *cluster1*, and subclasses *F clef (left)* and *C clef* have been added

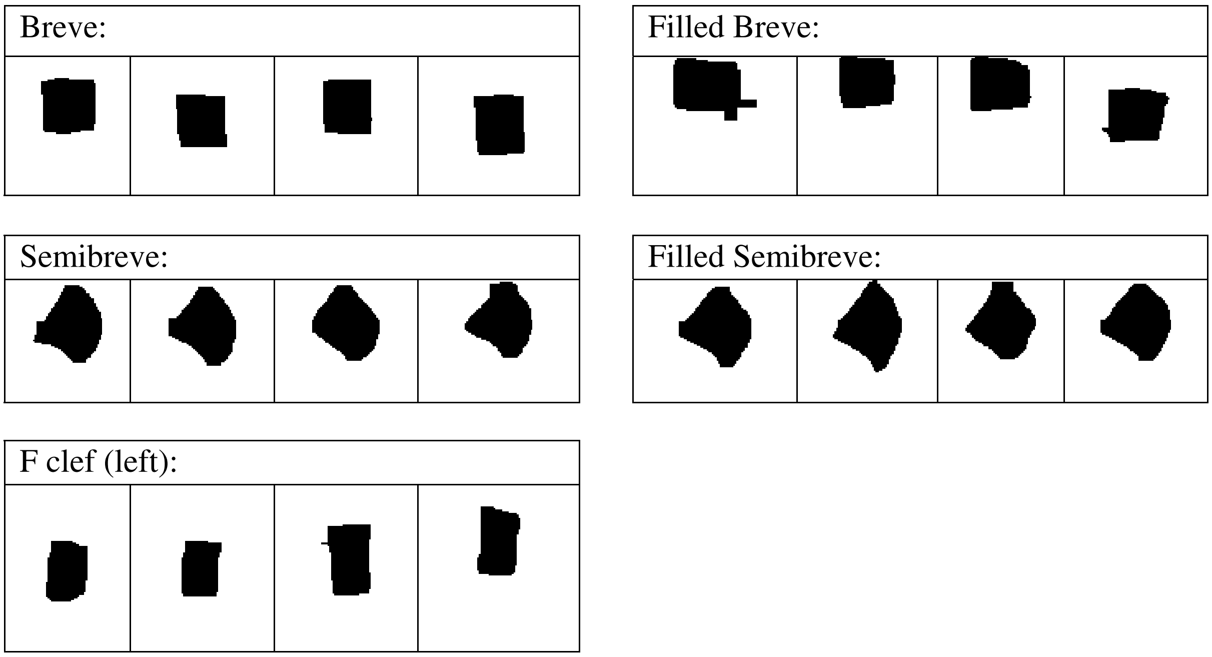


Fig. 11. Musical symbols of the class BB_NS_Box after vertical closure and horizontal open

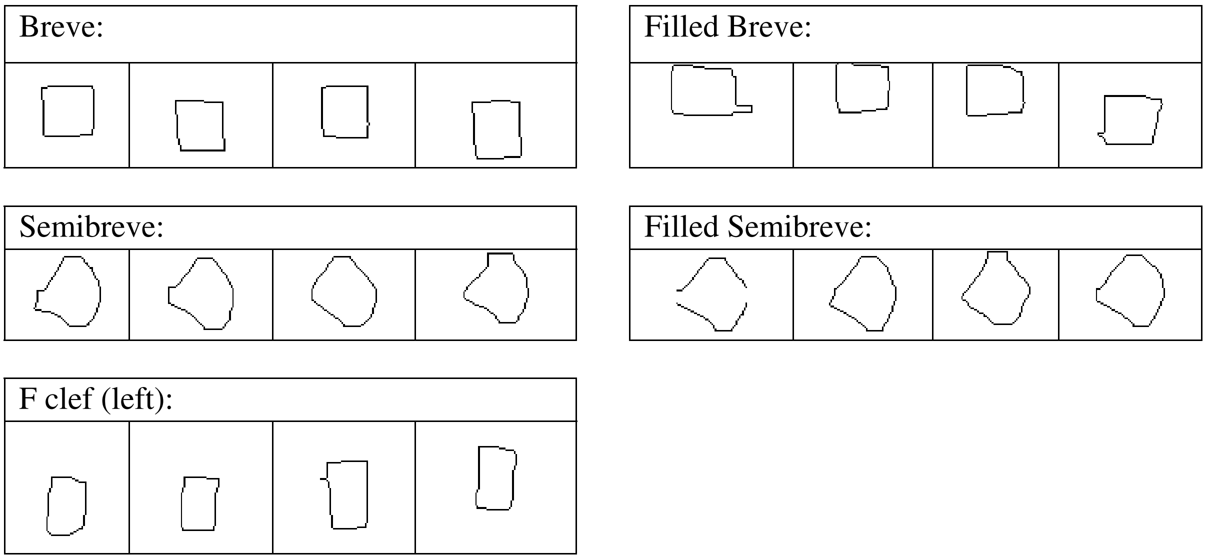


Fig. 12. Musical symbols of the class BB_NS_Box, after vertical closure, horizontal open, and contour evaluation

Table 7. Classification results for the BB_NS_Box problem

Situation	Training set	Test set	Accuracy (%) NP	Accuracy (%) MAP	Accuracy (%) k-NN ($k = 3$)
Training set	28 pages 1381 objects	28 pages 1381 objects	99.57	99.71	99.78
Half division	14 pages 723 objects	14 pages 658 objects	99.24	99.54	99.54
1-out	27 × 28 pages	1 × 28 pages	99.59	99.60	99.68

Table 8. Confusion matrices for the classifiers MAP, NP, and k-NN

MAP	Square	Losangle	NP	Square	Losangle	k-NN	Square	Losangle
Square	163	3	Square	165	1	Square	163	3
Losangle	0	492	Losangle	4	488	Losangle	0	492

to the superclass *cluster 4*. With these new classes, when an object is classified in cluster 1, it may also be a semibreve, and if it is classified in cluster 4, it may also be an F clef (left) or a C clef.

The final hierarchy can be considered a very good one since (1) the clusters are well separated, as can be seen in the cluster quality measures (Fig. 8); (2) the set of subclasses with more than one superclass is small (only *semibreve*, *F clef (left)* and *C clef* are divided); and (3) with the final hierarchy a simple MAP classifier achieves an accuracy of nearly 100% (eight misclassifications out of 1883; 99.58% accuracy).

4.2 Clustering a specific superclass

The work of subdividing the classes obtained in Sect. 4.1 was class-specific. In most cases, a promising feature or set of features was selected and tested, using that feature set with the two abovementioned classifiers and also the k-nearest neighbor (k-NN) [6]. Sometimes we recurred to image transformations, as morphological filtering [16], before feature calculation.

The superclass called BB_NS_Box, the clustering tests, the classifiers produced, and the classification tests are presented below. Figure 1 shows examples of original images of musical symbols belonging to the BB_NS_Box class.

The square-like objects were separated from the losangle-like objects, and the final class hierarchy of the class BB_NS_Box has two subclasses: BB_NSB_Square and BB_NSB_Losangle. In order to distinguish between them, some morphological transformations were performed on the images in Fig. 10. First, to close all objects with a hole in the middle, we used vertical closure with a structuring element of 25 pixels in one column. Then, to eliminate vertical dashes that are unnecessary for the separation, a horizontal opening with a structuring element of ten pixels in one line has been applied. The result of applying these transformations is shown in Fig. 11.

In order to distinguish between squares [the class containing the objects breve, filled breve, and F clef (left)] and losangles (the class with the objects semibreve and filled semibreve), a code based on the contour of the images is used. First, a morphological transformation is applied to find the contour. The result is depicted in Fig. 12.

The number of transitions of a given type found in the contour is used as a feature. Following the contour pixel by pixel, eight types of transitions can be found that consist of eight possible directions: left (0°), up (90°), up-left (45°), and so on. The directions can be reduced to four, since, for instance, 0° is similar to 180°. In general, square objects have mostly 0° and 90° transitions, while losangle objects have mostly 45° and 135° transitions. In this example, the feature consists of counting 45° and 135° transitions.

The quality measures for the two clusters, squares and losangles, are presented in Table 6, where the Manhattan distance measure presents slightly better results, and thus it is chosen as distance measure.

Classification tests for the three types of classifiers are shown in Table 7.

The accuracy of the three types of classifiers is presented for three different situations:

- Training set – the accuracy of the classifier on all the training sets, consisting of 28 pages.
- Half division – uses half of the total set of pages for training and the other half for testing.
- 1-out – performs 28 tests, using 27 pages for training and the remaining page for testing in each test. The test is performed for all 28 pages. The results shown are the combination of the 28 tests performed.

The confusion matrices for the three classifiers, MAP, NP, and k-NN, is presented in Table 8 for the half-division test case.

The MAP classifier is chosen because it gives better accuracy than the NP classifier, and the better accuracy of the k-NN classifier does not justify its greater complexity.

Table 9. Classification results for the ROMA problem

Situation	Training set	Test set	Accuracy (%) NP
Training set	28 pages 1883 objects	28 pages 1883 objects	98.23
Half division	14 pages 975 objects	14 pages 908 objects	96.74
1-out	27 × 28 pages	1 × 28 pages	97.00

4.3 Overall results

The global recognition system proposed in this paper was applied to the overall book MM37 studied under the ROMA project. Table 9 presents the results obtained for this problem. The situations are the same as in the BB_NS_Box case. The confusion matrix for the half-division test case is presented in Table 10.

These results show that a high accuracy of the recognition system designed can be obtained using the methodology proposed in this paper. The output of the system is a normalized music sheet with the original staff printed using straight staff lines and normalized musical

Table 10. Classification results for the global ROMA recognition system. A: Breve (gen), B: F clef (left), C: Filled breve, D: Filled semibreve, E: Semibreve (gen), F: Suspension top, G: F clef (right), H: Minim (tail up), I: Minim (tail down), J: C clef, K: F clef (total), L: Ligature (3 bre), M: Ligature (2 bre), N: Bar line, O: Dot, P: Rest (semibreve), Q: Rest (breve), R: Tail (breve), S: Flat, T: Finale, U: Ligature (5 bre)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
A	144	2	1	0	2	0	1	2	0	0	0	0	1	0	0	0	0	0	0	0	0
B	6	33	0	0	1	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0
C	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	497	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	45	0	0	0	2	0	0	0	0	0	0	0	0	0	0
H	0	1	0	0	0	0	0	19	0	0	2	0	0	0	0	0	0	0	0	0	0
I	0	1	0	0	0	0	0	0	16	0	0	0	1	0	0	0	0	0	0	0	0
J	1	0	0	0	0	0	1	0	0	13	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	5	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

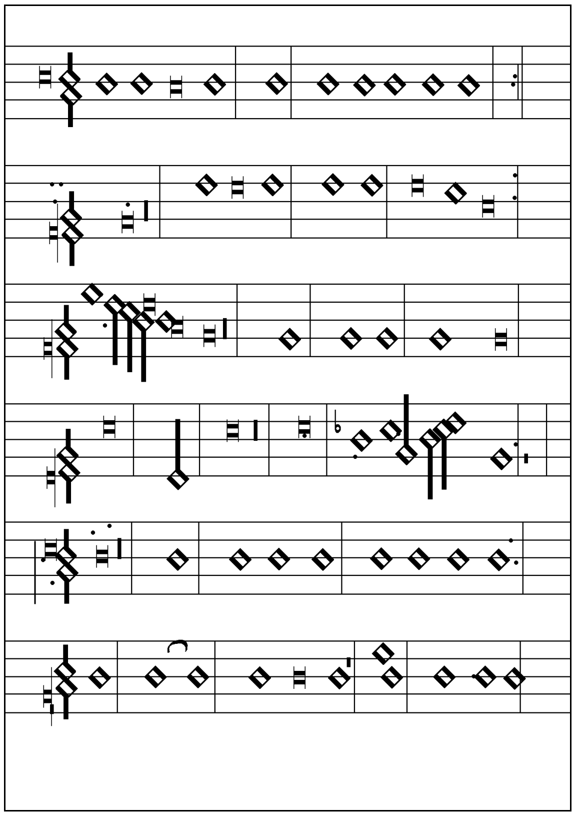


Fig. 13. Normalized music sheet

symbols. An example is shown in Fig. 13. For each class of musical symbol (figure), a normalized figure was set. When the system classifies an object in one class, that object is substituted by the normalized figure of its class in the normalized sheet image.

5 Conclusions

This paper proposes a graph-like classification method that was applied to the recognition of music in ancient manuscripts, characterized by a specific notation. The classification method includes an elaborate and domain-specific process of image segmentation and the construction of a class hierarchy associated with recognizers that distinguish between clusters of classes based on selected object features. A new method for the search of optimal graph hierarchy (manual and automated) and for the classification algorithms themselves has been proposed.

Future work includes the complete automation of recognition graph building, the generalization of the developed techniques to other notations, the development of other techniques to deal with specific difficulties of such notations, and the use of these systems to produce a digital library of ancient music.

Acknowledgements. We would like to thank BGUC (Biblioteca Geral da Universidade de Coimbra) and Prof. José Maria Pedrosa Cardoso for their contributions to this project. This project is supported by the Portuguese National Project

PRAXIS/C/EEI/12122/1998 and by the Programa de Financiamento Plurianual de unidades de I&D (POCTI), do Quadro Comunitário de Apoio III.

References

1. Armand J-P (1993) Musical score recognition: a hierarchical and recursive approach. Second international conference on document analysis and recognition, ICDAR, Tsukuba Science City, Japan
2. Bainbridge D, Carter N (1997) Automatic recognition of music notation. In: Bunke H, Wang P (eds) Handbook of optical character recognition and document image analysis, World Scientific, Singapore, pp 557–603
3. Banks S (1990) Signal processing, image processing and pattern recognition. Prentice-Hall, New York
4. Blostein D, Baird HS (1992) A critical survey of music image analysis. In: Baird HS, Bunke H, Yamamoto K (eds) Structured document image analysis, Springer, Berlin Heidelberg New York, pp 405–434
5. Pinto JC, Vieira P, Ramalho M, Mengucci M, Pina P, Muge F (2000) Ancient music recovery for digital libraries. Fourth European conference on research and advanced technology for digital libraries, ECDL 2000, Lisbon, pp 24–35
6. Duda RO, Hart PE (1973) Pattern recognition and scene analysis. Wiley, New York
7. Ferraté G, Pavlidis T, Sanfeliu A, Bunke H (eds) (1988) Syntactic and structural pattern recognition. NATO ASI Series, vol 45. Springer, Berlin Heidelberg New York
8. Hoppner F, Klawonn F, Kruse R, Runkler T (1999) Fuzzy cluster analysis: methods for classification, data analysis and image recognition. Wiley, New York
9. Jang JSR, Sun C-T, Mizutani E (1997) Neuro fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice-Hall, New York
10. Kohavi R, Provost F (1998) Glossary of terms. Editorial for the special issue on applications of machine learning and the knowledge discovery process, 30:271-274
11. Leplumey J, Camillerapp J, Lorette G (1993) A robust detector for music staves. Second international conference on document analysis and recognition, ICDAR, Tsukuba City Science, Japan
12. Ng KC, Boyle RD (1996) Recognition and reconstruction of primitives in music scores. Image Vision Comput 14:39–46
13. Patterson DW (1996) Artificial neural networks: theory and applications. Prentice-Hall, New York
14. Pépin F, Randriamahefa R, Fluhr C, Philipp S, Cocquerez JP (1993) Printed music recognition. Second international conference on document analysis and recognition, ICDAR, Tsukuba City Science, Japan
15. Rossant F (2002) A global method for music symbol recognition in typeset music sheets. Pattern Recogn Lett 23:1129–1141
16. Serra J (1982) Image analysis and mathematical morphology, vol 1. Academic, London
17. Yu FT, Jutamulia S (eds) Optical pattern recognition. Cambridge University Press, Cambridge
18. Wang PS (1977) Handbook of optical character recognition and document image analysis. World Scientific, Singapore

Caldas Pinto was born in Leiria, Portugal in 1951 and graduated from Instituto Superior Técnico, Lisbon in 1974. He received his PhD in control systems in Manchester in 1983. He is associate professor at the Instituto Superior Técnico. His research interests include image processing and pattern recognition, principally with respect to old documents, and vision-based control chiefly as it applies to robotics.

Pedro Vieira was born in Lisbon, Portugal in 1977. He graduated from Universidade Técnica de Lisboa, Portugal in 2000. He works in an information systems development company and conducts research in pattern recognition applied to document analysis and recognition at IDMEC/Instituto Superior Técnico.

João Miguel da Costa Sousa was born in 1966 in Lisbon, Portugal. He graduated from the Technical University of Lisbon in 1989 and received his M.Sc. degree in mechanical engineering from that institution in 1992. He received his PhD in electrical engineering from the Delft University of Technology, The Netherlands, in 1998. He is currently assistant professor at the Instituto Superior Técnico, Technical University of Lisbon. His main research interests include fuzzy-model-based control, intelligent control, optimization and fuzzy decision making, and classification. He is an associate editor of the IEEE Transactions on Fuzzy Systems.