25

Mining Digital Music Score Collections: Melody Extraction and Genre Recognition

Pedro J. Ponce de León, José M. Iñesta and David Rizo
Department of Software and Computing Systems
University of Alicante,
Spain

1. Introduction

In the field of computer music, pattern recognition algorithms are very relevant for music information retrieval (MIR) applications. Two challenging tasks in this area are the automatic recognition of musical genre and melody extraction, having a number of applications like indexing and selecting musical databases.

One of the main references for music is its melody. In a practical environment of digital music score collections the information can be found in standard MIDI file format. Music is structured as a number of tracks in this file format, usually one of them containing the melodic line, while other tracks contain the accompaniment. Finding that melody track is very useful for a number of applications, like speeding up melody matching when searching in MIDI databases, extracting motifs for musicological analysis, building music thumbnails or extracting melodic ringtones from MIDI files.

In the first part of this chapter, musical content information is modeled by computing global statistical descriptors from track content. These descriptors are the input to a random forest classifier that assigns the probability of being a melodic line to each track. The track with the highest probability is then selected as the one containing the melodic line of the MIDI file. The first part of this chapter ends with a discussion on results obtained from a number of databases of different music genres.

The second part of the chapter deals with the problem of classifying such melodies in a collection of music genres. A slightly different approach is used for this task, first dividing a melody track in segments of fixed length. Statistical features are extracted for each segment and used to classify them as one of several genres. The proposed methodology is presented, covering the feature extraction, feature selection, and genre classification stages. Different supervised classification methods, like Bayesian classifier and nearest neighbors are applied. As a proof of concept, the performance of such algorithms against different description models and parameters is analyzed for two particular musical genres, like jazz and classical music.

2. Symbolic music information retrieval

Music information retrieval (MIR) is a field of research devoted to the extraction of meaningful information from the content of music sources. In the application of pattern recognition techniques to MIR, two main folds can be found in the literature: audio information retrieval and symbolic music information retrieval.

Source: Pattern Recognition Techniques, Technology and Applications, Book edited by: Peng-Yeng Yin,
ISBN 978-953-7619-24-4, pp. 626, November 2008, I-Tech, Vienna, Austria

In the first case the raw digital audio signal is processed. Usually `wav` or `MP3` files are the input to these systems. No explicit information about notes, voices or any musical symbol or tag is encoded in the signal. On the other hand, symbolic `MIR` is based on processing symbols with direct musical meaning: notes with pitch and duration, lyrics, tags, etc. The most common formats used as input for these systems are ASCII text files like `kern`, `abc`, `MusicXML`, or binary files containing note control information like `MIDI` files. In these formats input data contain information about what and how is to be played, instead of the rendered music itself like in the audio signal. The semantics of both approaches is different, at least in the first stage of information retrieval algorithms. In the case of symbolic processing, as musical information use to be found as input, most existing music information theory can be applied to the task. On the other hand, the use as raw audio lacks from the basic music information as notes or voices notes or voices. Signal processing techniques must be used to extract this musical data, thus introducing noise to the actual musical material found in the audio. Currently, some of the most active tasks in audio information retrieval have as objective the extraction of that musical information like note onsets, timbre or voices. With this preprocessing of the raw audio, many of the work lines that can be found in the symbolic music information retrieval can also be tackled, but with the drawback of the possibly ill musical data extracted.

The goals of symbolic music information retrieval can be said to be more close to the actual music theory or musicological analysis that those of audio information retrieval. Some of the most active work areas in the symbolic approach nowadays is listed below:

- *Genre and mood classification:* the objective in those two tasks is to tell the mood or musical genre a given input belongs to (McKay & Fujinaga, 2004; Zhu et al., 2004; Cruz et al., 2003; Buzzanca, 2002; Pérez-Sancho et al., 2004; Dannenberg et al., 1997; van Kranenburg & Backer, 2004; Ponce de León et al., 2004)
- *Similarity and retrieval:* the final target in this work line is to be able to perform a search in a music data base to get the most similar pieces to an input query (Typke et al., 2003; Lemstrom & Tarhio, 2000)
- *Cover song identification:* the detection of plagiarisms and variations of the same song is the main goal in this case (Grachten et al., 2004; Li & Sleep, 2004; Rizo et al., 2008)
- *Key finding:* Guess the tonality and key changes of the score given the notes (Rizo et al., 2006a; Temperley, 2004)
- *Melody identification:* to identify the melody line among several MIDI tracks or music staffs, in opposite to those that contain accompaniment (Rizo et al., 2006b)
- *Motive extraction:* to find motives (short note sequences) in a score that are the most repeated ones acting as the main themes of the song (Serrano & Iñesta, 2006)
- *Meter detection:* given the input song reconstruct the meter from the flow of notes (Temperley, 2004)
- *Score segmentation:* split the song in parts like musical phrases (Spevak et al., 2002)
- *Music analysis:* perform musicological analysis for teaching, automatic or computed assisted composition, automatic expressive performance, and build a musical model for other MIR tasks (Illescas et al., 2007)

3. Melody characterization

A huge number of digital music score can be found on the Internet or in multimedia digital libraries. These scores are stored in files conforming to a proprietary or open format, like MIDI or the various XML music formats available. Most of these files contain music organized in a way such that the leading part of the music, the melody, is stored separately from the rest of the musical content, which is often the accompaniment for the melody. In particular, a standard MIDI file is usually structured as a number of tracks, one for each voice in a music piece. One of them usually contains a melodic line or *melody*, specially in the case of modern popular music.

Melody is a somewhat elusive musical term that often refers to a central part of a music piece that catches most of the listener's attention, and which the rest of music parts are subordinated to. This is one of many definitions that can be found in many places, particularly music theory manuals. Most of these definitions share some melody traits, like 'sequential', 'monophonic', 'main reference', 'unity in diversity', 'lyrics related', 'culturally dependent', etc.

Our goal is to automatically find this melody track in a MIDI file using statistical properties of the musical content and pattern recognition techniques. The proposed methodology can be applied to other symbolic music file formats, because the information used to take decisions is based solely on how the notes are arranged within each voice of a digital score. Only the feature extraction front-end would need to be adapted for dealing with other formats.

The identification of the melody track is very useful for a number of applications. For example, in melody matching, when the query is either in symbolic format (Uitdenbogerd & Zobel, 1999) or in audio format (Ghias et al., 1995), the process can be speeded up if the melody track is known or if there is a way to know which tracks are most likely to contain the melody, because the query is almost always a melody fragment. Another useful application can be helping motif extraction systems to build music thumbnails of digital scores for music collection indexing.

3.1 Related works

To our best knowledge, the automatic description of a melody has not been tackled as a main objective in the literature. The most similar problem to the automatic melody definition is that of extracting a melody line from a polyphonic source. This problem has been approached from at least three different points of view with different understandings of what a melody is. The first approach is the extraction of melody from a polyphonic *audio* source. For this task it is important to describe the melody in order to leave out those notes that are not candidates to belong to the melody line (Eggink & Brown, 2004). In the second approach, a melody line (mainly monophonic) must be extracted from a *symbolic* polyphonic source where no notion of *track* is used (I.Karydis et al., 2007). With this approach, Uitdenbogerd and Zobel (Uitdenbogerd & Zobel, 1998) developed four algorithms for detecting the melodic line in polyphonic MIDI files, assuming that a melodic line is a monophonic sequence of notes. These algorithms are based mainly on note pitches; for example, keeping at every time the note of highest pitch from those that sound at that time (skyline algorithm).

Other works on this line focus on how to split a polyphonic source into a number of monophonic sequences by partitioning it into a set of melodies (Marsden, 1992). In general, these works are called monophonic reduction techniques (Lemstrom & Tarhio, 2000).

The last approach to melody characterization is to select one track containing the melody from a list of input tracks of symbolic polyphonic music (e.g. MIDI). This is, by the way, our own approach. Other authors, like (Ghias et al., 1995), built a system to process MIDI files extracting a sort of melodic line using simple heuristics. (Tang et al., 2000) presented a work where the aim was to propose candidate melody tracks, given a MIDI file. They take decisions based on single features derived from informal assumptions about what a melody track may be. (Madsen & Widmer, 2007) try to solve the problem by the use of several combination of the entropies of different melody properties like pitch classes, intervals, etc.

3.2 What's a melody?

Before focusing on the machine learning methodology to extract automatically the characterization of a *melody*, the musical concept ofmelody needs to be reviewed.

Melody is a concept that has been given many definitions, all of them complementary. The variability of the descriptions can give an idea on the difficulty of the task to extract a description automatically.

From the music theory point of view, Ernst Toch (Toch, 1997) defines it as "*a succession of different pitch sounds brighten up by the rhythm*". He also writes "*a melody is a sound sequence with different pitches, in opposition to its simultaneous audition that constitutes what is named as chord*". He distinguishes also the term 'melody' from the term 'theme'.

A music dictionary (Sadie & Grove, 1984) defines melody as: "*a combination of a pitch series and a rhythm having a clearly defined shape*".

The music theory literature lacks works about melody in favour of works about counterpoint, harmony, or "form" (Selfridge-Field, 1998). Besides, the concept of melody is dependant on the genre or the cultural convention. The most interesting studies about melody have appeared in recent years, mainly influenced by new emerging models like generative grammars (Baroni, 1978), artificial intelligence (Cope, 1996), and Gestalt and cognitive psychology (Narmour, 1990). All these works place effort on understand the melody in order to generate it automatically.

The types of tracks and descriptions of *melody* versus *accompaniment* is posed in (Selfridge-Field, 1998). The author distinguishes:

- *compound* melodies where there is only a melodic line where some notes are principal, and others tend to accompany, being this case the most frequent in unaccompanied string music.
- *self-accompanying* melodies, where some pitches pertain both to the thematic idea and to the harmonic (or rhythmic) support
- *submerged* melodies consigned to inner voices
- *roving* melodies, in which the theme migrates from part to part
- *distributed* melodies, in which the defining notes are divided between parts and the prototype cannot be isolated in a single part.

From the audio processing community, several definitions can be found about what a melody is. Maybe, the most general definition is found in (Kim et al., 2000): "*melody is an auditory object that emerges from a series of transformations along the six dimensions: pitch, tempo, timbre, loudness, spatial location, and reverberant environment*".

(Gomez et al., 2003) gave a list of mid and low-level features to describe melodies:

- Melodic attributes derived from numerical analysis of pitch information: number of notes, tessitura, interval distribution, melodic profile, melodic density.
- Melodic attributes derived from musical analysis of the pitch data: key information, scale type information, cadence information.
- Melodic attributes derived from a structural analysis: motive analysis, repetitions, patterns location, phrase segmentation.

Another attempt to describe a melody can be found in (Temperley, 2004). In that book, Temperley proposes a model of melody perception based on three principles:

- Melodies tend to remain within a narrow pitch range.
- Note-to-note intervals within a melody tend to be small.
- Notes tend to conform to a key profile (a distribution) that depends on the key.

All these different properties a melody should have can be used as a reference to build an automatic melody characterization system.

4. Melody track identification

As stated before, in this work the aim is to decide which of the tracks contains the main melody in a multitrack standard MIDI file. For this, we need to assume that the melody is indeed contained in a single track. This is the case for today's world music.

The features that should characterize melody and accompaniment voices must be defined in order to be able to select the melodic track. There are some features in a melody track that, at first sight, seem to be enough for identifying it, like the presence of higher pitches (Uitdenbogerd & Zobel, 1998) or being monophonic. Unfortunately, any empirical analysis will show that these hypotheses do not hold in general, and more sophisticated criteria need to be devised in order to take accurate decisions.

To overcome these problems, a classifier ensemble able to learn what is a melodic track from note distribution statistics has been used in this work. In order to setup and test the classifier, a number of data sets based on several music genres and consisting of multitrack standard MIDI files have been constructed. All tracks in such files are labeled either as melody or non-melody.

The classic methodology in the pattern recognition field has been used in this work. A vector of numeric descriptors is extracted from each track of a target midifile, and these descriptors are the input to a classifier that assigns to each track its probability of being a melody. This is the big picture of the method. The random forest classifier (Breiman, 2001) – an ensemble of decision trees– was chosen as the pattern recognition tool for this task. The WEKA (Witten & Frank, 1999) toolkit was used to implement the system.

4.1 MIDI track characterization

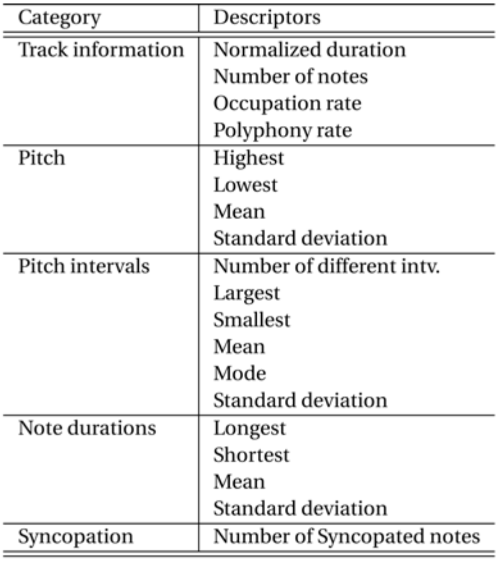
The content of each non-empty track[1] is characterized by a vector of descriptors based on descriptive statistics of note pitches and durations that summarize track content information. This kind of statistical description of musical content is sometimes referred to as *shallow structure description* (Pickens, 2001; Ponce de León et al., 2004b).

A set of descriptors has been defined, based on several categories of features that assess melodic and rhythmic properties of a music sequence, as well as track related properties. This set of descriptors is presented in Table 1. The left column indicates the category being analyzed, and the right one shows the kind of statistics describing properties from that category.

Four features were designed to describe the track as a whole and fifteen to describe particular aspects of its content. For these fifteen descriptors, both normalized and non-normalized versions have been computed. The former were calculated using the formula $(value_i - min)/(max - min)$, where $value_i$ is the descriptor to be normalized corresponding to

[1] tracks containing at least one note event. Empty tracks are discarded.

the *i*-th track, and *min* and *max* are, respectively, the minimum and maximum values for this descriptor for all the tracks of the target midifile. This allows to know these properties proportionally to the other tracks in the same file. This way, a total number of 4+15×2 = 34 descriptors were initially computed for each track.



Category	Descriptors
Track information	Normalized duration Number of notes Occupation rate Polyphony rate
Pitch	Highest Lowest Mean Standard deviation
Pitch intervals	Number of different intv. Largest Smallest Mean Mode Standard deviation
Note durations	Longest Shortest Mean Standard deviation
Syncopation	Number of Syncopated notes

Table 1. Track content descriptors

The track information descriptors are its normalized duration, number of notes, occupation rate (proportion of the track length occupied by notes), and the polyphony rate, defined as the ratio between the number of ticks in the track where two or more notes are active simultaneously and the track duration in ticks.

Pitch descriptors are measured using MIDI pitch values. The maximum possible MIDI pitch is 127 (note G_8) and the minimum is 0 (note C_{-2}). The interval descriptors summarize information about the difference in pitch between consecutive notes. The absolute pitch interval values were computed. Finally, note duration descriptors were computed in terms of beats, so they are independent from the MIDI file resolution.

4.2 Feature selection

The descriptors listed above are a complete list of computed features, but any pattern recognition system needs to explore which are those features that actually are able to separate the target classes.

Some descriptors show evidence of statistically significant differences when comparing their distributions for melody and non-melody tracks, while other descriptors do not. This property is implicitly observed by the classification technique utilized (see Section 4.3), that performs a selection of features in order to take decisions.

A view to the graphs in Figure 1 provides some hints on how a melody track could look like. This way, a melody track seems to have less notes than other non-melody tracks, an average mean pitch, it contains small intervals, and has not too long notes. When this sort of hints are combined by the classifier, a decision about the track "melodicity" is taken.

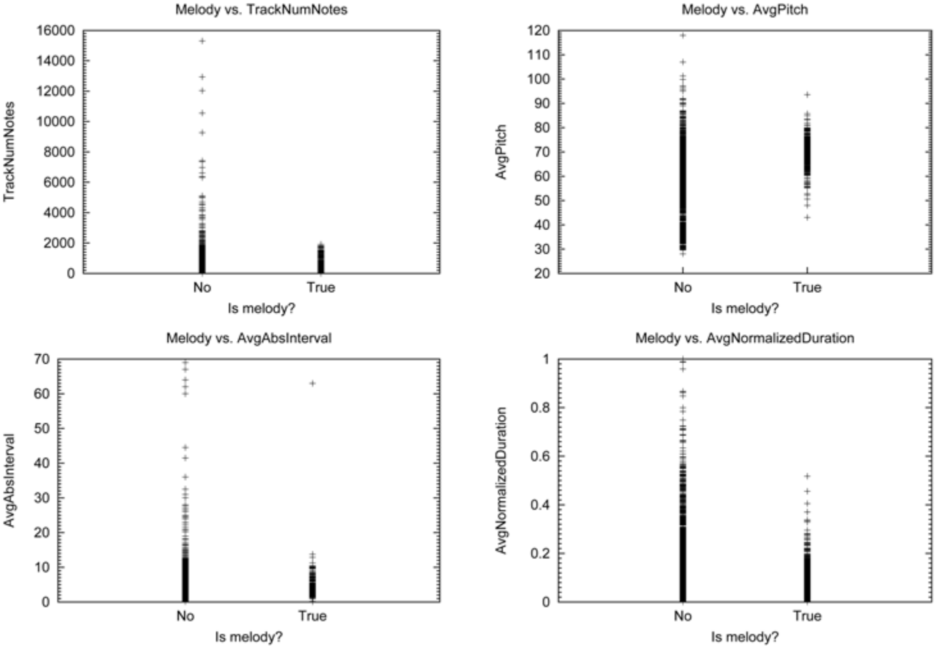


Fig. 1. Distribution of values for some descriptors: (top-left) number of notes, (top-right) mean pitch, (bottom-left) mean absolute interval, and (bottom-right) mean relative duration.

4.3 The random forest classifier

A number of classifiers were tested in an initial stage of this research and the random forest classifier yielded the best results among them, so it was chosen for the experiments presented in the next section.

Random forests (Breiman, 2001) are weighed combinations of decision trees that use a random selection of features to build the decision taken at each node. This classifier has shown good performance compared to other classifier ensembles with a high robustness with respect to noise. One forest consists of *K* trees. Each tree is built to maximum size using CART (Duda et al., 2000) methodology without pruning. Therefore, each leaf on the tree corresponds to a single class. The number *F* of randomly selected features to split on the training set at each node is fixed for all the trees. After the trees have grown, new samples are classified by each tree and their results are combined, giving as a result a membership probability for each class.

In our case, the membership for class "melody" is interpreted as the probability that a track will contain a melodic line.

4.4 Track selection procedure

There are MIDI files that contain more than one track which is suitable to be classified as melody: singing voice, instrument solos, melodic introductions, etc. On the other hand, as usually happens in classical music, some songs do not have an obvious melody, like in complex symphonies or single-track piano sequences. The algorithm proposed here can deal with the first case. For the second case, there are more suitable methods (Uitdenbogerd & Zobel, 1998) that perform melody extraction from polyphonic data.

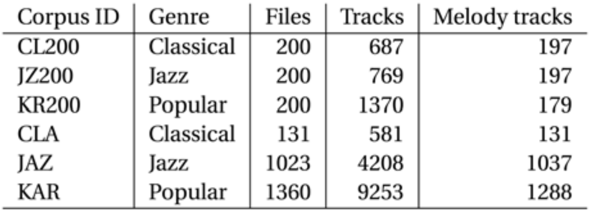
In some of the experiments in the next section, at most one melody track per MIDI file is selected. However, a file can contain more than one melody track. Therefore, given a file, all its non-empty tracks are classified and their probabilities of being a melody are obtained. Then the track with the highest probability is selected as the melody track. If all tracks have near-zero probability (actually less than 0.01), no melody track is selected –that is, all tracks are considered as non-melody tracks.

In the first stages of this work, a probability threshold around 0.5 was established in order to discard tracks whose probability of being a melody was below that value. This resulted in some files in our test datasets being tagged as melody-less. However most of those files actually have a melody. In general, this produced systems with lower estimated accuracy than systems with a near-zero probability threshold.

4.5 Experiments

4.5.1 Datasets and tools

Six corpora (see Table 2) were created, due to the lack of existing databases for this task. The files were downloaded from a number of freely accessible Internet sites. First, three corpora (named JZ200, CL200, and KR200) were created to set up the system and to tune the parameter values. JZ200 contains jazz music files, CL200 has classical music pieces where there was a melody track, and KR200 contains popular music songs with a part to be sung (karaoke (.kar) format). All of them are made up of 200 files. Then, three other corpora (named JAZ, CLA, and KAR) from the same music genres were compiled from a number of different sources to validate our method. This dataset is available for research purposes on request to the authors.



Corpus ID	Genre	Files	Tracks	Melody tracks
CL200	Classical	200	687	197
JZ200	Jazz	200	769	197
KR200	Popular	200	1370	179
CLA	Classical	131	581	131
JAZ	Jazz	1023	4208	1037
KAR	Popular	1360	9253	1288

Table 2. Corpora used in the experiments, with identifier, music genre, number of files, total number of tracks, total number of melody tracks and baseline success ratio.

The main difficulty for building the data sets was to label the tracks in the MIDI files. Text tagging of MIDI tracks based on metadata such as the track name, is unreliable. Thus, a manual labeling approach was carried out. A musician listened to each one of the MIDI files playing all tracks simultaneously. For each file, tracks containing the perceived melody were identified and tagged as *melody*. The rest of tracks in the same file were tagged as *non-melody*. In particular, introduction passages, second voices or instrumental solo parts were tagged as *non-melody*.

Some songs had no tracks tagged as melody because either it was absent, or the song contained some kind of melody-less accompaniment, or it had a canon-like structure, where the melody moves constantly from one track to another. Other songs contained more than one melody track (e.g. duplicates, often with a different timbre) and all those tracks were tagged as *melody*.

The WEKA package was used to carry out the experiments described here. It was extended to compute the proposed track descriptors directly from MIDI files.

Four experiments have been carried out, as listed below:

- Melody vs. non-melody classification
- Melody track selection
- Genre specificity
- Training set specificity

The first one tries to assess the capability of random forests to classify melodic and non-melody tracks properly. In the second experiment, the aim is to evaluate how accurate the system is for identifying the melody track in a MIDI file. Finally, the specificity of the system with respect to both the music genre and the corpora utilized were tested.

4.5.2 Melody versus non-melody classification

As described before, our aim is to assess the capability of the classifier to discriminate melody from non-melody tracks. Therefore, given a set of tracks, this experiment classifies them either as melody or non-melody. The random forest classifier assigns a class membership probability to each test sample, so in this experiment a test track is assigned to the class with the highest membership probability.

As a proof of concept, three independent sub-experiments were carried out, using the three 200-file corpora (CL200, JZ200, and KR200). This way, 2826 tracks provided by these files were classified in two classes: *melody* / *non-melody*. A ten-fold cross-validation scheme was used to estimate the accuracy of the method. The success results are shown in Table 3 and figure 2, along with the baseline ratio when considering a dumb classifier that always output the most frequent class (*non-melody* for all datasets). The remarkable success percentages obtained are due to the fact that the classifier was able to successfully map the input feature vector space to the class space. This shows that content statistics in combination with decision tree based learning can produce good results on the task at hand.

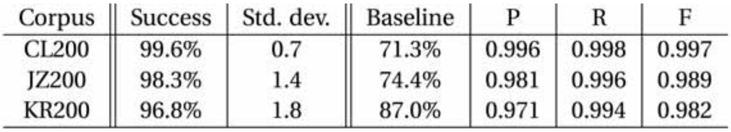
Also, precision (P), recall (R) and the F-measure (F) are shown for melody tracks. These standard information retrieval measures are based on the so-called *true-positive* (TP), *false-positive* (FP) and *false-negative* (FN) counts. For this experiment, TP is the number of melody tracks successfully classified, FP is the number of misclassified non-melody tracks, and finally, FN is the number of misclassified melody tracks. The precision, recall and F-measure are calculated as follows:

$$P = \frac{TP}{TP+FP}$$

$$R = \frac{TP}{TP+FN}$$

$$F = \frac{2 \times R \times P}{R+P}$$

These results have been obtained using K = 10 trees and F = 5 randomly selected features for the random forest trees. The same classifier structure was used in the rest of experiments presented in the next sections.



Corpus	Success	Std. dev.	Baseline	P	R	F
CL200	99.6%	0.7	71.3%	0.996	0.998	0.997
JZ200	98.3%	1.4	74.4%	0.981	0.996	0.989
KR200	96.8%	1.8	87.0%	0.971	0.994	0.982

Table 3. Melody versus non-melody classification results.

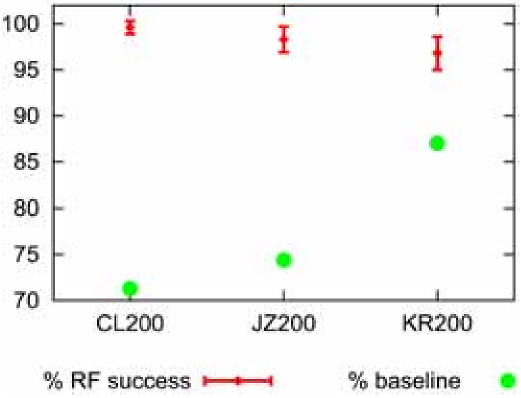


Fig. 2. Melody vs. non-melody classification success and baseline

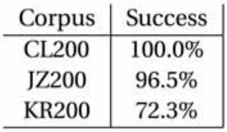
4.5.3 Melodic track selection experiment

In this second experiment, the goal is to test wether the method selects the proper melody track from a MIDI file. For this experiment, the system was trained the same way as in the previous one, but now a test sample is not a single track but a MIDI file. Due to the limited number of samples available (200 per corpus), this experiment was performed using a leave-one-out scheme at the MIDI file level to estimate the classification accuracy. The classifier assigns a class membership probability to each track in a test file. For each file, the system outputs the track number that gets highest membership probability for class *melody*, except when all these probabilities are near-zero, in which case the system considers the file has no melody track.

The classifier answer for a givenMIDI file is considered correct if

1. At least one track is tagged as *melody* and the selected track is one of them.
2. There are no melody tracks and the classifier outputs no melody track number.

Results are shown in Table 4.



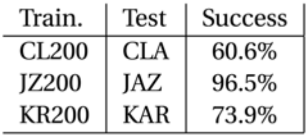
Corpus	Success
CL200	100.0%
JZ200	96.5%
KR200	72.3%

Table 4. Melody track selection results.

Note the high quality of the results for CL200 and JZ200. However, a lower success rate has been obtained for the karaoke files. This is due to the fact that 31 out of 200 files in this corpus were tagged by a human expert as having no actual melody track, but they have some portions of tracks that could be considered as melody (like short instrument solo parts), thus confusing the classifier as FP hits, therefore lowering the classifier precision for this corpus.

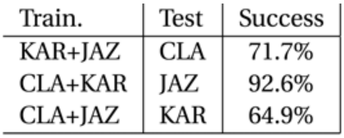
4.5.4 Genre specificity

This experiment was designed in order to evaluate the system robustness against different corpora. In particular, it is interesting to know how specific the classifier's inferred rules are with respect to the music genre of files considered for training. For it, two melody track selection sub-experiments, like the ones in the previous section, were performed: in the first one, the classifier was trained with a 200-file corpus of a given music genre, and tested with a different corpus of the same genre (see Table 5). For the second sub-experiment, the classifier was trained using data from two genres and then tested with files from the third genre dataset (see Table 6).



Train.	Test	Success
CL200	CLA	60.6%
JZ200	JAZ	96.5%
KR200	KAR	73.9%

Table 5. Melody track selection within genre.



Train.	Test	Success
KAR+JAZ	CLA	71.7%
CLA+KAR	JAZ	92.6%
CLA+JAZ	KAR	64.9%

Table 6. Melody track selection across genres.

The results in Table 5 show that the performance of the system degrades when more complex files are tested. The 200-file corpora are datasets that include MIDI files that were selected among many others for having an 'easily' (for a human) identifiable melody track. This holds also for the JAZ corpus, as most jazz music MIDI files have a lead voice (or instrument) track plus some accompaniment tracks like piano, bass and drums. However, it does not hold in general for the other two corpora. Classical music MIDI files (CLA corpus) come in very different structural layouts, due to both the way that the original score is organized and the idiosyncrasy of the MIDI file authors. This is also mostly true for the KAR corpus. Moreover, karaoke files tend to make intensive use of duplicate voices and dense pop arrangements with lots of tracks containing many ornamentation motifs. In addition, we have verified the presence of very short sequences for the CLAS corpus, causing less quality in the statistics that also degrades the classification results.

As both the training and test corpora contain samples of the same music genre, better results were expected. However, the CLA and KAR corpora are definitively harder to deal with, as it became clear in the second experiment presented in this section. So, it can be said that the difficulty of the task resides more on the particular internal organization of tracks in the MIDI files than on the file music genre, despite that the results in Table 5 seem to point out that genre makes a difference. The second experiment presented in Table 6 showed some evidence in that direction.

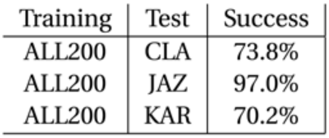
Most errors for the CLA test set were produce because a non-melody track was selected as melody (a kind of false positive). Same type of errors can be found for the KAR corpus, along with errors due to the classifier not finding any melody tracks in files with a melody tagged track (a kind of false-negative).

Results from the second sub-experiment show that performance is poorer (with respect to the first one) when no data from the test genre were used for training. This does not happen in classical music, probably due to effects related to the problems expressed above.

4.5.5 Training set specificity

To see how conditioned are these results by the particular training sets utilized, a generalization study was carried out building a new training set merging the three 200-files corpora (named *ALL200*), and then using the other corpora for test. The problem to solve is again the one discussed in section 4.5.3: selecting the proper melody track from a MIDI file. The results are detailed in Table 7.

This shows that, when using a multi-genre dataset, the performance of the system is somewhat improved (now the training set contains samples from the same genre as the test dataset). Note that the results are better despite that the size of the training set is smaller than the size of those used in Section 4.5.4.



Training	Test	Success
ALL200	CLA	73.8%
ALL200	JAZ	97.0%
ALL200	KAR	70.2%

Table 7. Melody track selection by genres when training with data from all the genres.

When combining all the success results, taking into account the different cardinalities of the test sets, the average successful melody track identification percentage is 81.2 %.

The method proposed here can be used as a tool for extracting the melody track in conjunction with a system for music genre recognition presented in the next section. This system is a melody based genre recognition system. It extracts information from melody tracks in order to recognize the melody genre. This way MIDI files need not to be preprocessed by an expert in order to identify the melody track.

5. Music genre recognition

One of the problems to solve in MIR is the modelization of music genre. The computer could be trained to recognise the main features that characterise music genres in order to look for that kind of music over large musical databases. The same scheme is suitable to learn stylistic features of composers or even model a musical taste for users. Other application of such a system can be its use in cooperation with automatic composition algorithms to guide this process according to a given stylistic profile.

A number of papers explore the capabilities of machine learning methods to recognise music genre. Pampalk et al. (Pampalk et al., 2003) use self-organising maps (SOM) to pose the problem of organising music digital libraries according to sound features of musical themes, in such a way that similar themes are clustered, performing a content-based classification of the sounds. (Whitman et al., 2001) present a system based on neural networks and support vector machines able to classify an audio fragment into a given list of sources or artists. Also, in (Soltau et al., 1998) a neural system to recognise music types from sound inputs is described. An *emergent* approach to genre classification is used in (Pachet et al., 2001), where a classification emerges from the data without any *a priori* given set of genres. The authors use co-ocurrence techniques to automatically extract musical similarity between titles or artists. The sources used for classification are radio programs and databases of compilation CDs.

Other works use music data in symbolic form (most MIDI data) to perform genre recognition. (Dannenberg et al., 1997) use a naive Bayes classifier, a linear classifier and neural networks to recognize up to eight moods (genres) of music, such as lyrical, frantic, etc. Thirteen statistical features derived from MIDI data are used for this genre discrimination. In (Tzanetakis et al., 2003), pitch features are extracted both from MIDI data and audio data and used separately to classify music within five genres. Pitch histograms regarding to the tonal pitch are used in (Thom, 2000) to describe blues fragments of the saxophonist Charlie Parker. Also pitch histograms and SOM are used in (Toiviainen & Eerola, 2001) for musicological analysis of folk songs. Other researchers use sequence processing techniques like Hidden Markov Models (Chai & Vercoe, 2001) and universal compression algorithms (Dubnov & Assayag, 2002) to classify musical sequences.

(Stamatatos & Widmer, 2002) use stylistic performance features and the discriminant analysis technique to obtain an ensemble of simple classifiers that work together to recognize the most likely music performer of a piece given a set of skilled candidate pianists. The input data are obtained from a computer-monitored piano, capable of measuring every key and pedal movement with high precision.

Compositions from five well known eighteenth-century composers are classified in (van Kranenburg & Backer, 2004) using several supervised learning methods and twenty genre features, most of them being counterpoint characteristics. This work offers some conclusions about the differences between composers discovered by the different learning methods.

In other work (Cruz et al., 2003), the authors show the ability of grammatical inference methods for modeling musical genre. A stochastic grammar for each musical genre is inferred from examples, and those grammars are used to parse and classify new melodies. The authors also discuss about the encoding schemes that can be used to achieve the best recognition result. Other approaches like multi-layer feed-forward neural networks (Buzzanca, 2002) have been used to classify musical genre from symbolic sources.

(McKay & Fujinaga, 2004, 2007a) use low and mid-level statistics of MIDI file content to perform music genre recognition by means of genetic algorithms and pattern recognition techniques. They have developed several tools for feature extraction from music symbolic sources (particularly MIDI files) or web sites (McKay & Fujinaga, 2006a, 2007b). In (McKay & Fujinaga, 2006b), the authors provide some insight on why is it worth continuing research in automatic music genre recognition, despite the fact that the ground-truth information available for research is often not too reliable, being subject to subjective tagging, market forces or being culture-dependent. Most of the classification problems detected seem to be related to the lack of reliable ground-truth, from the definition of realistic and diverse genre labels, to the need of combining features of different nature, like cultural, high- and low-level features. They also identify, in particular, the need for being able to label different sections of a music piece with different tags.

The system presented in this section share some features with the one developed by McKay, as the use of low level statistics and pattern recognition techniques but, while McKay extract features from the MIDI file as a whole, our system focus on melody tracks, using a *sliding* *window* technique to obtain melody segments that become instances to feed the pattern recognition tools. This allows to obtain partial decisions for a melody track that can offer the users sensible information for different parts of a music work. Also, this decisions can be combined to output a classification decision for a music piece.

5.1 An experimental framework for automatic music genre recognition

In this section a framework for experimenting on automatic music genre recognition from symbolic representation of melodies (digital scores) is presented. It is based on shallow structural features of melodic content, like melodic, harmonic, and rhythmic statistical descriptors. This framework involves all the usual stages in a pattern recognition system, like feature extraction, feature selection, and classification stages, in such a way that new features and corpora from different musical genres can be easily incorporated and tested.

Our working hypothesis is that melodies from a same musical genre may share some common low-level features, permitting a suitable pattern recognition system, based on statistical descriptors, to assign the proper musical genre to them.

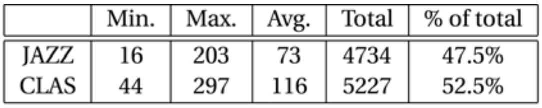
Two well-defined music genres, like jazz and classical, have been chosen as a workbench for this research. The initial results have been encouraging (Ponce de León & Iñesta, 2003) but the method performance for different classification algorithms, descriptor models, and parameter values needed to be thoroughly tested. This way, a framework for musical genre recognition can be set up, where new features and new musical genres can be easily incorporated and tested.

This section presents the proposed methodology, describing the musical data, the descriptors, and the classifiers used. The initial set of descriptors will be analyzed to test their contribution to the musical genre separability. These procedures will permit us to build reduced models, discarding not useful descriptors. Then, the classification results obtained with each classifier and an analysis of them with respect to the different description parameters will be presented. Finally, conclusions and possible lines of further work are discussed.

5.2 Musical data

MIDI files from jazz and classical music, were collected. These genres were chosen due to the general agreement in the musicology community about their definition and limits. Classical melody samples were taken from works by Mozart, Bach, Schubert, Chopin, Grieg, Vivaldi, Schumann, Brahms, Beethoven, Dvorak, Haendel, Paganini and Mendelssohn. Jazz music samples were standard tunes from a variety of well known jazz authors including Charlie Parker, Duke Ellington, Bill Evans, Miles Davis, etc. The MIDI files are composed of several tracks, one of them being the melody track from which the input data are extracted[2]. The corpus is made up of a total of 110 MIDI files, 45 of them being classical music and 65 being jazz music. The length of the corpus is around 10000 bars (more than 6 hours of music). Table 8 summarizes the distribution of bars from each genre. This dataset is available for research purposes on request to the authors.

[2] All the melodies are written in 4/4 meter. Anyway, any other meter could be used because the measure structure is not use in any descriptor computation. All the melodies are monophonic sequences (at most one note is playing at any given time).



	Min.	Max.	Avg.	Total	% of total
JAZZ	16	203	73	4734	47.5%
CLAS	44	297	116	5227	52.5%

Table 8. Distribution of melody length in bars

This corpus has been manually checked for the presence and correctness of key, tempo and meter meta-events, as well as the presence of a monophonic melody track. The original conditions under which the MIDI files were created are unknown; They may be human performed tracks or sequenced tracks (i.e. generated fromscores) or even something of both worlds. Nevertheless, most of the MIDI files seem to fit a rather common scheme: a human-performed melody track with several sequenced accompaniment tracks.

The monophonic melodies consist of a sequence of musical events that can be either notes or silences. The pitch of each note can take a value from 0 to 127, encoded together with the MIDI note onset event. Each of these events at time *t* has a corresponding note off event at time *t* +*d*, being *d* the note duration measured in ticks[3]. Time gaps between a note off event and the next note onset event are silences.

5.3 Description scheme

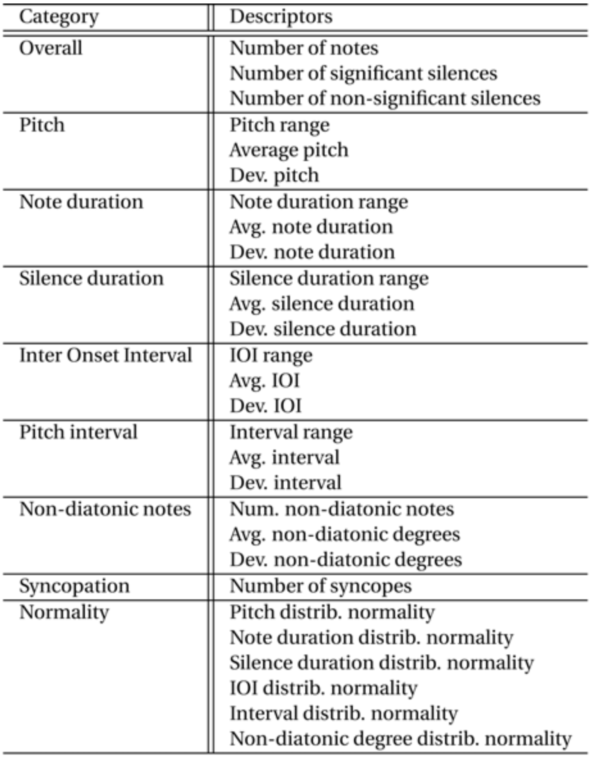
A description scheme has been designed based on descriptive statistics that summaris the content of the melody in terms of pitches, intervals, durations, silences, harmonicity, rhythm, etc.

Each sample is a vector of musical descriptors computed from each melody segment available (See section 5.4 for a discussion about how these segments are obtained). Each vector is labeled with the genre of the melody which the segment belongs to. We have defined an initial set of descriptors based on a number of feature categories that assess the melodic, harmonic and rhythmic properties of a musical segment, respectively.

This initial model is made up of 28 descriptors summarized in table 9, and described next:

- Overall descriptors:
  - *Number of notes*, *number of significant silences, and number of not significant silences*. The adjective *significant* stands for silences explicitly written in the underlying score of the melody. In MIDI files, short gaps between consecutive notes may appear due to interpretation nuances like *stacatto*. These gaps (interpretation silences) are not considered significant silences since they should not appear in the score. To make a distinction between kinds of silence is not possible from the MIDI file and it has been made based on the definition of a silence duration threshold. This value has been empirically set to a duration of a sixteenth note. All silences with longer or equal duration than this threshold are considered significant.
- Pitch descriptors:
  - *Pitch range* (the difference in semitones between the highest and the lowest note in the melody segment), *average pitch* relative to the lowest pitch, and *standard deviation of pitches* (provides information about how the notes are distributed in the score).

[3] A *tick* is the basic unit of time in a MIDI file and is defined by the resolution of the file, measured in ticks per beat.



Category	Descriptors
Overall	Number of notes Number of significant silences Number of non-significant silences
Pitch	Pitch range Average pitch Dev. pitch
Note duration	Note duration range Avg. note duration Dev. note duration
Silence duration	Silence duration range Avg. silence duration Dev. silence duration
Inter Onset Interval	IOI range Avg. IOI Dev. IOI
Pitch interval	Interval range Avg. interval Dev. interval
Non-diatonic notes	Num. non-diatonic notes Avg. non-diatonic degrees Dev. non-diatonic degrees
Syncopation	Number of syncopes
Normality	Pitch distrib. normality Note duration distrib. normality Silence duration distrib. normality IOI distrib. normality Interval distrib. normality Non-diatonic degree distrib. normality

Table 9. Musical descriptors

- Note duration, silence duration and IOI[4] descriptors are measured in ticks and computed using a time resolution of Q = 48 ticks per bar [5]. Interval descriptors are computed as the difference in absolute value between the pitches of two consecutive notes.
- Harmonic descriptors:
  - *Number of non diatonic notes.* An indication of frequent excursions outside the song key (extracted from the MIDI file) or modulations.
  - *Average degree of non diatonic notes.* Describes the kind of excursions. This degree is a number between 0 and 4 that indexes the non diatonic notes of the diatonic scale of the tune key, that can be major or minor key[6]
  - *Standard deviation of degrees of non diatonic notes.* Indicates a higher variety in the non diatonic notes.

[4] An IOI is the distance, in ticks, between the onsets of two consecutive notes. Two notes are considered consecutive even in the presence of a silence between them.

[5] This is call quantisation. Q = 48 means that when a bar is composed of 4 beats, each beat can be divided, at most, into 12 ticks.

[6] Non diatonic degrees are: 0: ♭II, 1: ♭III (♮III for minor key), 2: ♭V, 3: ♭VI, 4: ♭VII. The key is encoded at the beginning of the melody track. It has been manually checked for correctness in our data.

- Rhythmic descriptor:
  - *Number of syncopations*: notes that do not begin at measure beats but in some places between them (usually in the middle) and that extend across beats.
- Normality descriptors. They are computed using the D'Agostino statistic for assessing the distribution normality of the n values v_i in the segment for pitches, durations, intervals, etc. The test is performed using this equation:

$$D = \frac{\sum_i (i - \frac{n+1}{2}) v_i}{\sqrt{n^3 (\sum_i v_i^2 - \frac{1}{n}(\sum_i v_i)^2)}} \tag{1}$$

For pitch and interval properties, the range descriptors are computed as maximum minus minimum values, and the average-relative descriptors are computed as the average value minus the minimum value (only considering the notes in the segment). For durations (note duration, silence duration, and IOI descriptors) the range descriptors are computed as the ratio between the maximum and minimum values, and the average-relative descriptors are computed as the ratio between the average value and the minimum value.

This descriptive statistics is similar to histogram-based descriptions used by other authors (Thom, 2000; Toiviainen & Eerola, 2001) that also try to model the distribution of musical events in a music fragment. Computing the range, mean, and standard deviation from the distribution of musical properties, we reduce the number of features needed (each histogram may be made up of tens of features). Other authors have also used this sort of descriptors to classify music (Tzanetakis et al., 2003; Blackburn, 2000), mainly focusing on pitches.

5.4 Free parameter space

Given a melody track, the statistical descriptors presented above are computed from equal length segments extracted from the track, by defining a window of size ω measures. Once the descriptors of a segment have been extracted, the window is shifted δ measures forward to obtain the next segment to be described. Given a melody with $m > 0$ measures, the number of segments s of size $\omega > 0$ obtained from that melody is

$$s = \begin{cases} 1 & \text{if } \omega \geq m \\ 1 + \lceil \frac{m-\omega}{\delta} \rceil & \text{otherwise} \end{cases} \tag{2}$$

showing that at least one segment is extracted in any case (ω and s are positive integers; m and δ may be positive fractional numbers).

Taking ω and δ as free parameters in our methodology, different datasets of segments have been derived from a number of values for those parameters. The goal is to investigate how the combination of these parameters influences the segment classification results. The exploration space for this parameters will be referred to as $\omega\delta$-space. A point in this space is denoted as $\langle\omega, \delta\rangle$.

ω is the most important parameter in this framework, as it determines the amount of information available for the descriptor computations. Small values for ω would produce windows containing few notes, providing little reliable statistical descriptors. Large values for ω would lead to merge –probably different– parts of a melody into a single window and they also produce datasets with fewer samples for training the classifiers (see Eq. 2). The value of δ would affect mainly to the number of samples in a dataset. A small δ value combined with quite large values for *ω* may produce datasets with a large number of samples (see also Eq. 2). The details about the values used for these parameters can be found in section 5.7.

5.5 Feature selection procedure

The features described above have been designed according to those used in musicological studies, but there is no theoretical support for their genre classification capability. We have applied a selection procedure in order to keep those descriptors that better contribute to the classification. The method assumes feature independence, that is not true in general, but it tests the separability provided by each descriptor independently, and uses this separability to obtain a descriptor ranking.

Consider that the *M* descriptors are random variables $\{X_j\}_{j=1}^{M}$ whose *N* sample values are those of a dataset corresponding to a given *ω*δ-point. We drop the subindex *j* for clarity, because all the discussion applies to each descriptor. We split the set of *N* values for each descriptor into two subsets: $\{X_{C,I}\}_{i=1}^{N_C}$ are the descriptor values for classical samples and $\{X_{J,I}\}_{i=1}^{N_J}$ are those for the jazz samples, being N_C and N_J the number of classical and jazz samples, respectively. X_C and X_J are assumed to be independent random variables, since both sets of values are computed from different sets of melodies. We want to know whether these random variables belong to the same distribution or not. We have considered that both sets of values hold normality conditions, and assuming that the variances for X_C and X_J are different in general, the test contrasts the null hypothesis $H_0 \equiv \mu_C = \mu_J$ against $H_1 \equiv \mu_C \neq \mu_J$. If H_1 is concluded, it is an indication that there is a clear separation between the values of this descriptor for the two classes, so it is a good feature for genre classification. Otherwise, it does not seem to provide separability between the classes.

The following statistical for sample separation has been applied:

$$z = \frac{|\bar{X}_C - \bar{X}_J|}{\sqrt{\frac{s_C^2}{N_C} + \frac{s_J^2}{N_J}}} \quad , \tag{3}$$

where $\overline{X}_C$ and $\overline{X}_J$ are the means, and s_C^2 and s_J^2 the variances for the descriptor values for both classes. The greater the *z* value is, the wider the separation between both sets of values is for that descriptor. A threshold to decide when H_0 is more likely than H_1, that is to say, the descriptor passes the test for the given dataset, must be established. This threshold, computed from a t-student distribution with infinite degrees of freedom and a 99.7% confidence interval, is *z* = 2.97. Furthermore, the *z* value permits to arrange the descriptors according to their separation ability.

When this test is performed on a number of different *ω*δ-point datasets, a threshold on the number of passed tests can be set as a criterion to select descriptors. This threshold is expressed as a minimum percentage of tests passed. Once the descriptors are selected, a second criterion for grouping them permits to build several descriptor models incrementally. First, selected descriptors are ranked according to their *z* value averaged over all tests. Second, descriptors with similar *z* values in the ranking are grouped together. This way, several descriptor groups are formed, and new descriptor models can be formed by incrementally combining these groups. See the section 5.7.1 for the models that have been obtained.

5.6 Classifier implementation and tuning

Two algorithms from different classification paradigms have been used for genre recognition. They are fully supervised methods: The Bayesian classifier and the *k*-nearest neighbor (*k*-NN) classifier (Duda et al., 2000).

The Bayesian classifier is parametric and, when applied to a two-class problem, computes a discriminant function:

$$g(X) = log\frac{P(X \mid \omega_1)}{P(X \mid \omega_2)} + log\frac{\pi_1}{\pi_2} \tag{4}$$

for a test sample X where $P(X \mid \omega_i)$ is the conditional probability density function for class i and π_i are the priors of each class. Gaussian probability density functions for each genre are assumed for each descriptor. Means and variances are estimated separately for each class from the training data. The classifier assigns a sample to ω_1 if $g(X) > 0$, and to ω_2 otherwise. The decision boundaries, where $g(X) = 0$, are in general hyperquadrics in the feature space.

The *k*-NN classifier uses an Euclidean metrics to compute the distance between the test sample and the training samples. The genre label is assigned to the test sample by a majority decision among the nearest *k* training samples (the *k*-neighborhood).

5.7 Experiments and results on music genre recognition

5.7.1 Feature selection results

The feature selection test presented in section 5.5 has been applied to datasets corresponding to 100 randomly selected points of the $\omega\delta$-space. This is motivated by the fact that the descriptor computation is different for each ω and the set of values is different for each δ, so the best descriptors may be different for different $\omega\delta$-points. Thus, by choosing a set of such points, the sensitivity of the classification to the feature selection procedure can be analysed. Being a random set of points is a good trade-off decision to minimise the risk of biasing this analysis.

The descriptors were sorted according to the average z value ($\overline{z}$) computed for the descriptors in the tests. The list of sorted descriptors is shown in table 10. The $\overline{z}$ values for all the tests and the percentage of passed tests for each descriptor are displayed. In order to select descriptors, a threshold on the number of passed tests has been set to 95%. This way, those descriptors which failed the separability hypothesis in more than a 5% of the experiments were discarded from the reduced models. Only 12 descriptors out of 28 were selected. In the right most column, the reduced models in which the descriptors were included are presented. Each model is denoted with the number of descriptors included in it.

Three reduced size models have been chosen, with 6, 10, and 12 descriptors. This models are built according to the $\overline{z}$ value as displayed in figure 3. The biggest gaps in the $\overline{z}$ values for the sorted descriptors led us to group the descriptors in these three reduced models. Note also that the values for $\overline{z}$ show a small deviation, showing that the descriptor separability is quite stable in the $\omega\delta$-space.

It is interesting to remark that at least one descriptor from each category of those defined in section 5.3 were selected for a reduced model. The best represented categories were pitches and intervals, suggesting that the pitches of the notes and the relation among them are the most influent features for this problem. From the statistical point of view, standard deviations were the most important features, since five from six possible ones were selected.

descriptor	$\bar{z}$	passed tests	models
Number of notes	22.5	100%	6,10,12
Average pitch	22.3	100%	6,10,12
Pitch range	22.2	100%	6,10,12
Interval range	20.3	100%	6,10,12
Syncopation	19.6	100%	6,10,12
Dev. pitch	18.7	100%	6,10,12
number of significant silences	14.2	100%	10,12
Interval distrib. normality	14.2	100%	10,12
Dev. interval	14.0	100%	10,12
Dev. IOI	13.2	97%	10,12
Dev. note duration	9.3	95%	12
Dev. non-diatonic degrees	9.1	100%	12
Dev. silence duration	6.3	94%	–
Silence duration range	6.1	87%	–
Note duration distrib. normality	6.0	89%	–
Avg. note duration	5.6	71%	–
Avg. silence duration	5.1	85%	–
Avg. non-diatonic degrees	4.9	66%	–
IOI range	4.7	53%	–
number of non-significant silences	4.5	76%	–
Silence duration distrib. normality	4.3	45%	–
Avg. IOI	4.2	53%	–
Non-diatonic degree distrib. normality	3.5	39%	–
Note duration range	3.3	34%	–
Pitch distrib. normality	2.6	25%	–
Num. non-diatonic notes	2.5	32%	–
IOI distrib. normality	2.2	20%	–
Avg. interval	1.7	14%	–

Table 10. Feature selection results

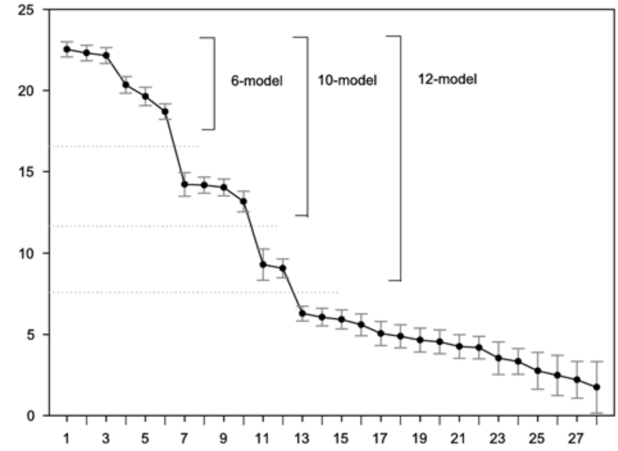


Fig. 3. Values for $\overline{z}$ for each descriptor as a function of their order numbers. The relative deviations for $\overline{z}$ in all the experiments are also displayed. The biggest gaps for $\overline{z}$ and the models are outlined.

5.7.2 The *ωδ*-space framework

The melodic segment parameter space has been established as follows:

$$\omega = 1, \ldots, 100 \tag{5}$$

and, for each *ω*

$$\delta = \begin{cases} 1, \ldots, \omega & \text{if } \omega \leq 50 \\ 1, \ldots, 20 & \text{otherwise} \end{cases} \tag{6}$$

The range for δ when *ω*> 50 has been limited to 20 due to the very few number of samples obtained with large δ values for this *ω* range. This setup produces a total of 2275 points ⟨*ω*,δ⟩ in the *ω*δ-space. A number of experiments have been made for each of these points: one with each classifier (Bayes, NN) for each of the four description models discussed in section 5.7.1. Therefore, 12 different experiments for each *ω*δ-point have been made, denoted by (*ω*,δ,*μ*,*γ*), where *μ* ∈ {6,10,12,28} is the description model and *γ* ∈ {Bayes,NN} the classifier used.

In order to obtain reliable results, a ten-fold crossvalidation scheme has been carried out for each of the (*ω*,δ,*μ*,*γ*) experiments, making 10 sub-experiments with about 10% of samples saved for test in each sub-experiment. The success rate for each (*ω*,δ,*μ*,*γ*) experiment is averaged for the 10 sub-experiments.

The partitions were made at the MIDI file level, to make sure that training and test sets do not share segments from any common melody. Also the partitions were made in such a way that the relative number of measures for both genres were equal to those for the whole training set. This permits us to estimate the prior probabilities for both genres once and then use them for all the sub-experiments. Once the partitions have been made, segments of *ω* measures are extracted from the melody tracks, and labeled training and test datasets containing *μ*-dimensional descriptor vectors are constructed.

To summarise, 27300 experiments consisting of 10 sub-experiments each, have been carried out. The maximum number of segments extracted is *s* = 9339 for the *ω*δ-point ⟨3,1⟩. The maximum for *s* is not located at ⟨1,1⟩ as expected, due to the fact that segments not containing at least two notes are discarded. The minimum is *s* = 203 for ⟨100,20⟩. The average number of segments in the whole *ω*δ-space is 906. The average proportion of jazz segments is 36% of the total number of segments, with a standard deviation of about 4%. This is a consequence of the classical MIDI files having a greater length in average than jazz files, although there are less classical files than jazz files.

5.8 Classification results

Each (*ω*,δ,*μ*,*γ*) experiment has an average success rate, obtained from the crossvalidation scheme discussed in the previous section. The results presented here are based on those rates.

5.8.1 Bayes classifier

For one sub-experiment in a point in the *ω*δ-space, all the parameters needed to train the Bayesian classifier are estimated from the particular training set, except for the priors of each genre, that are estimated from the whole set, as explained above.

Figure 4 shows the classification results for the Bayesian classifier over the *ω*δ-space for the 12-descriptor model. This was one of the best combination of model and classifier (89.5%of success) in average for all the experiments. The best results for this classifier were found around ⟨58,1⟩, where a 93.2% average success was achieved.

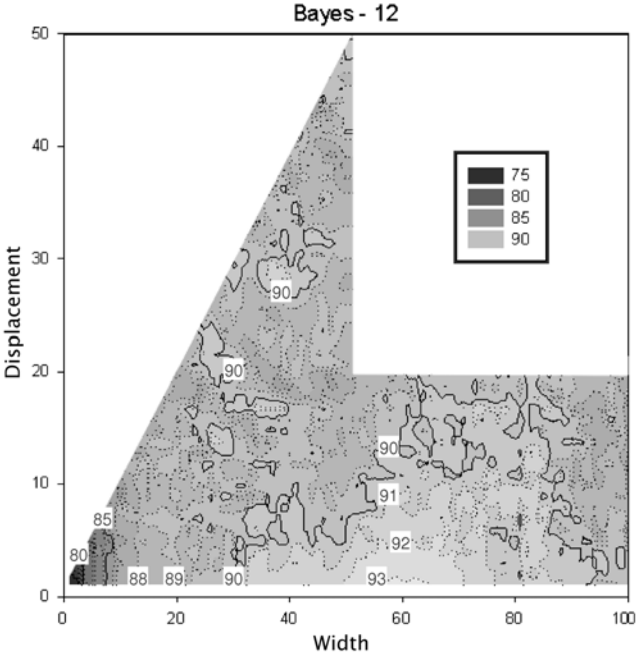


Fig. 4. Illustration of the recognition percentage in the ωδ-space for the Bayesian classifier with the 12-descriptor model. Numbers on top of level curves indicate the recognition percentage at places on the curve. The best results (around 93.2%) are found in the lighter area, with large widths and small displacements.

The best results for genre classification were expected to be found for moderate ω values, where enough musical events to calculate reliable statistical descriptors are contained in a segment, while musical events located in other parts of the melody are not mixed in a single segment. But the best results are generally obtained with a combination of large ω values and small δ. Experiments for $\omega = \infty$ (taking the whole melody as a single segment) are discussed in section 5.8.3.

The worst results occurred for small ω, due to the few musical events at hand when extracting a statistical description for such a small segment, leading to non reliable descriptors for the training samples.

All the three reduced models outperformed the 28-descriptor model (see Fig. 5 for a comparison between models for $\delta = 1$), except for $\omega \in [20,30]$, where the 28- descriptor model obtains similar results for small values of δ. For some reason, still unknown, the particular combination of ω and δ values in this range results in a distribution of descriptor values in the training sets that favours this classifier.

The overall best result (95.5% of average success) for the Bayesian classifier has been obtained with the 10-descriptor model in the point $\langle 98,1 \rangle$. See Table 11 for a summary of best results – indices represent the $\langle \omega, \delta \rangle$ values for which the best success rates were obtained. About 5% of the sub-experiments (4556 out of 91000) for all models yielded a 100% classification success.

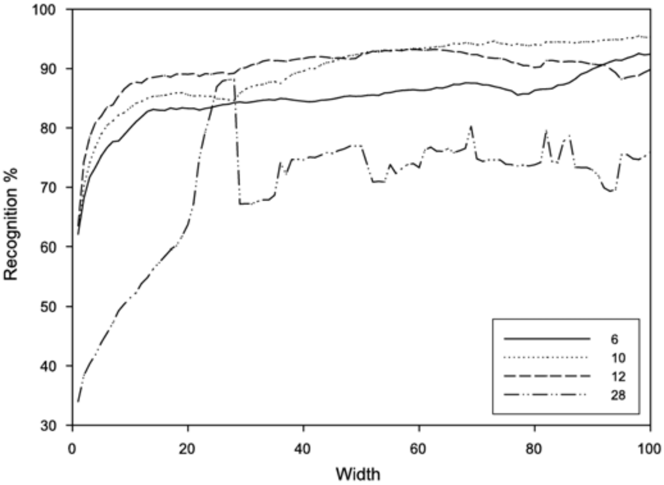


Fig. 5. Bayes recognition results for the differentmodels versus the windowwidth, with a fixed δ = 1.

model	Bayes	NN
6	$93.2_{\langle 100,2\rangle}$	$94.0_{\langle 91,16\rangle}$
10	$95.5_{\langle 98,1\rangle}$	$92.6_{\langle 99,19\rangle}$
12	$93.2_{\langle 58,1\rangle}$	$92.6_{\langle 98,19\rangle}$
28	$89.5_{\langle 41,33\rangle}$	$96.4_{\langle 95,13\rangle}$

Table 11. Best success rates

5.8.2 k-NN classifier

Before performing the main experiments for this classifier, a study of the evolution of the classification as a function of *k* has been designed, in order to test the influence of this parameter in the classification task. The results are displayed in Fig. 6. Recognition percentage is averaged for all ⟨*ω*,1⟩ points. Note that there is almost no variation in the recognition rate as *k* increases, except a small improvement for the 6-descriptor model. Thus, the simplest classifier was selected: *k* = 1, to avoid unnecessary time consumption due to the very large number of experiments to be performed.

Once the classifier has been set, the results for the different models were obtained and are displayed in Fig. 7 for δ = 1. All models performed comparatively for *ω* ≤ 35. For *ω* > 35, the 28-descriptor model begins to perform better than the reduced models. Its relatively high dimensionality and a greater dispersion in the samples (the larger the *ω*, the higher the probability of different musical parts to be contained in the same segment) causes larger distances among the samples, making the classification task easier for the *k*-NN.

The best results (96.4%) were obtained for the point ⟨95,13⟩ with the 28-descriptor model. The best results for all the models have been consistently obtained with very large segment lengths (see Table 11). The percentage of perfect (100%) classification sub-experiments amounts to 18.7% (17060 out of 91000).

For the whole *ω*δ-space, the NN classifier obtained an 89.2% in average with the 28-descriptor model, while the other models yielded similar rates, around 87%. The behavior of the 10- and 12-descriptor models was almost identical over the parameter space (Fig. 7) and for the different tested values for *k* (Fig. 6).

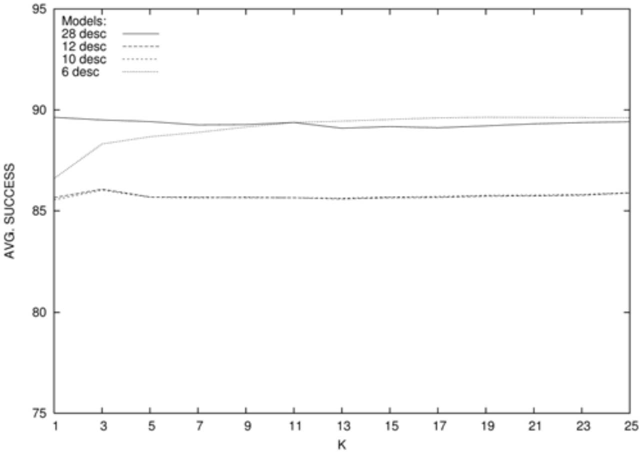


Fig. 6. Evolution of *k*-NN recognition for the different models against values of *k*.

model	Bayes	NN
6	84.2 ± 2.0	87.4 ± 2.9
10	88.5 ± 3.2	86.9 ± 2.5
12	89.5 ± 1.7	87.1 ± 2.5
28	71.1 ± 6.3	89.2 ± 4.5

Table 12. Averages and standard deviations of success rates

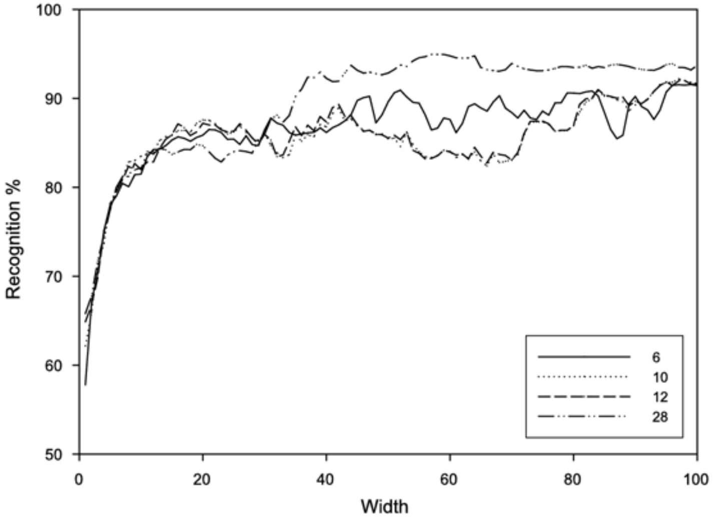
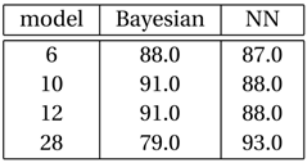


Fig. 7. NN recognition results for the different models versus the window width, with a fixed δ = 1.

5.8.3 Whole melody segment classification

The good results obtained for large *ω* called our attention to the question of how good would be the results of classifying whole melodies, instead of fragments, as presented so far. The first problem is the small number of samples available this way (110 samples for training and test). The results of these experiments are displayed in Table 13. The same 10-fold cross-validation scheme described in section 5.7.2 was used here. The results are comparable or even better than the average in the *ω*δ-space for both classification paradigms.



model	Bayesian	NN
6	88.0	87.0
10	91.0	88.0
12	91.0	88.0
28	79.0	93.0

Table 13. Average success rates for whole melody segment length (*ω* = ∞)

In spite of this good behavior for Bayes and *k*-NN, this approach has a number of disadvantages. Training is always more difficult due to the smaller number of samples. The classification cannot be performed *on-line* in a real-time system, because all the piece is needed in order to take the decision. There are also improvements to the presented methodology, like cooperative decisions using different segment classifications that can not be applied to the complete melody approach.

5.8.4 Results comparison

Bayesian and NN classifier performed comparatively. There were, in general, lower differences in average recognition percentages between models for NN than those found with the Bayesian classifier (see Table 12), probably due to its non-parametric nature.

An ANOVA test with Bonferroni procedure for multiple comparison statistics (Hancock & Klockars, 1996) was used to determine which combination of model and classifier gave the best classification results in average. According to this test, with the number of experiments performed, the required difference between any two recognition rates in Table 12 must be at least 0.45123 in order to be considered statistically different at the 95% confidence level. Thus, it can be stated that Bayes classifier with 12-descriptor model and NN classifier with 28-descriptor model perform comparatively well, and both outperform the rest of classifier and model combinations. The Bayes classifier has the advantage of using a reduced size description model.

In a recent work using the same data set (Pérez-Sancho et al., 2004), several text categorization algorithms have been used to perform genre recognition from whole melodies. In particular, a naive Bayes classifier with several multivariate Bernoulli and multinomial models are applied to binary vectors indicating the presence or absence of *n*-length words (sequences of *n* notes) in a melody. The work reported around 93% of success as the best performance. This is roughly the same best result reported here for the whole melody, although it is outperformed by the window classification results.

Results for the *ω*δ-space are hardly comparable with results by other authors, due to our use of segments instead of complete melodies, and mainly due to the different datasets put under study by different authors. Nevertheless a comparison attempt can be made with the results found in (Tzanetakis et al., 2003) for pair-wise genre classification. The authors use information from all the tracks on the MIDI files except tracks playing on the percussion channel. In that work, a 94% accuracy for Irish Folk music and Jazz identification is reported as the best result. Unfortunately, they did not use Classical samples. This accuracy percentage is similar to our results with whole melody length segments and the NN classifier (93%). A study on the classification accuracy as a function of the input data length is also reported, showing a behavior similar to the one reported here: classification accuracy using statistical information reaches its maximum for larger segment lengths, as they reported a maximum accuracy for five classes with 4 minute segment length. Our best results were obtained for $\omega > 90$ (see Table 11).

6. Some conclusions and future work

6.1 Conclusions on melody characterization

The method proposed here identifies the voice containing the melody in a multitrack digital score. It has been applied to standard MIDI files in which music is stored in several tracks, so the system determines whether a track is a melodic line or not. The track with the highest probability among the melodic tracks is finally labeled as the one containing the melody of that song.

The decisions are taken by a pattern recognition algorithm based on statistical descriptors (pitches, intervals, durations and lengths), extracted from each track of the target file. The classifier used for the experiments was a decision tree ensemble classifier named random forest. It was trained using MIDI tracks with the melody track previously labeled by a human expert.

The experiments yielded promising results using databases from different music genres, like jazz, classical, and popular music. Unfortunately, the results could not be compared to other systems because of the lack of similar works.

The results show that enough training data of each genre are needed in order to successfully characterize the melody track, due to the specificities of melody and accompaniment in each genre. Classical music is particularly hard for this task, because of the lack of a single track that corresponds to the whole melodic line in some files. In these files, melody moves from one track to another, as different instruments take the melody lead role. To overcome this problem, more sophisticated schemes oriented to melodic segmentation are needed.

The use of information about the layout of the tracks within a MIDI file is being investigated. We hope this would help to improve the performance of the system when dealing with particularly hard instances like the ones found in karaoke files. The extraction of human-readable rules from the trees in the random forest that help characterize melody tracks has been another topic of research that yielded some promising results. Several rule systems, including some fuzzy rule systems have been obtained (Ponce de León et al., 2007; Ponce de León et al., 2008). Being able to automatically obtain melody characterization rules that easily understandable by humans could be of interest for musicologists and would help building better tools for searching and indexing symbolically encoded music.

6.2 Conclusions on music genre recognition

Our main goal in this work has been to test the capability of melodic, harmonic, and rhythmic statistical descriptors to perform musical genre recognition. We have developed a framework for feature extraction, selection and classification experiments, where new corpora, description models, and classifiers can be easily incorporated and tested.

We have shown the ability of two classifiers, based on different paradigms, to map symbolic representations of melodic segments into a set of musical genres. Jazz and classical music have been used as an initial benchmark to test this ability. The experiments have been carried out over a parameter space defined by the size of segments extracted from melody tracks of MIDI files and the displacement between segments sequentially extracted from the same source. A total of 273000 classification sub-experiments have been performed.

From the feature selection stage, a number of interesting conclusions can be drawn. From the musical point of view, pitches and intervals have shown to be the most discriminant features. Other important features have been the number of notes and the rhythm syncopation. Although the former set of descriptors may be probably important in other genre classification problems, probably these latter two have found their importance in this particular problem of classical versus jazz. From the statistical point of view, standard deviations were very relevant, since five of them from six possible ones were selected.

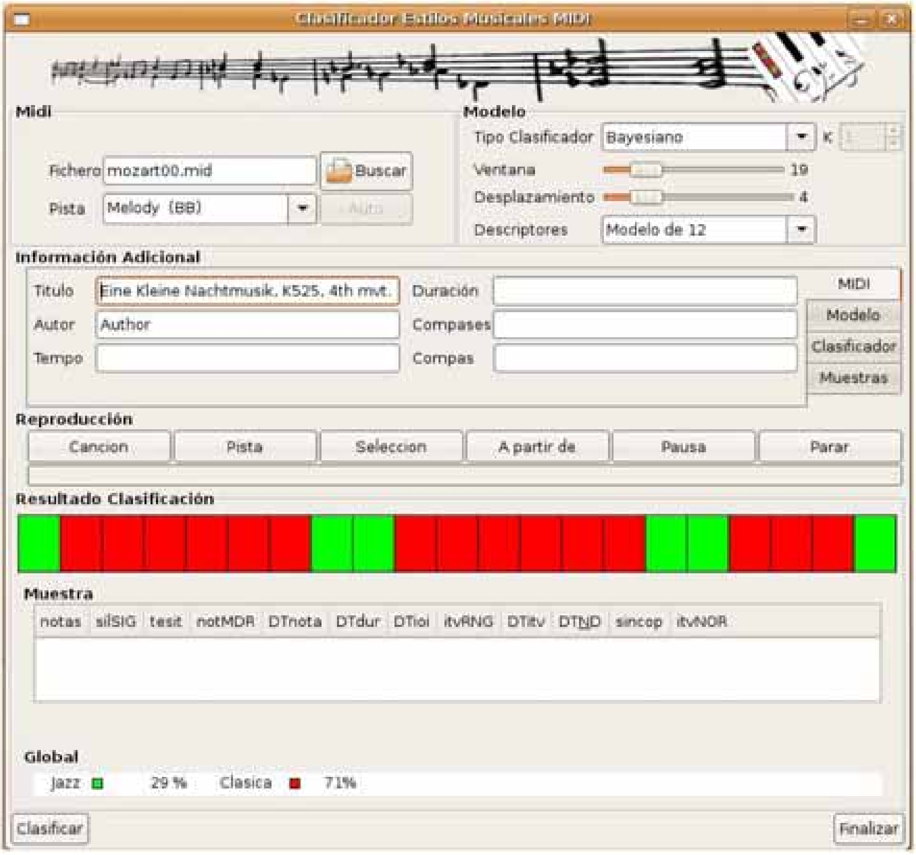
The general behavior for all the models and classifiers against the values for ω was to have bad classification percentages (around 60%) for $\omega = 1$, rapidly increasing to an 80% for $\omega \approx 10$, and then keep stable around a 90% for $\omega > 30$. This general trend supports the importance of describing large melody segments to obtain good classification results. The preferred values for δ were small, because they provide a higher number of training data.

Bayes and NN performed comparatively. The parametric approach preferred the reduced models but NN performed well with all models. In particular, with the complete model, without feature selection, it achieved very good rates, probably favored by the large distances among prototypes obtained with such a high dimensionality. The best average recognition rate in the $\omega\delta$-space has been found with the Bayesian classifier and the 12-descriptor model (89.5%), although the best result was obtained with the NN, that reached a 96.4% with $\omega = 95$ and $\delta = 13$.

Also, whole melody classification experiments were carried out, removing the segment extraction and segment classification stage. This approach is simpler, faster, and provide comparative results even with few training samples, but has a number of disadvantages. It does not permit the use of on-line implementations where the system can input data and take decisions in real-time, since all the piece needs to be entered to the classifier in a single step. In addition, the segment classification approach permits to analyse a long theme by sections, performing local classifications.

An extension to this framework is under development, where a voting scheme for segments is used to collaborate in the classification of the whole melody. The framework permits the training of a large number of classifiers that, combined in a multi-classifier system, could produce even better results.

An experimental graphical user interface has been developed to facilitate working on the problem of music genre recognition. The main motivation for such a tool is to allow investigate why classification errors occur. A snapshot of the interface (actually in spanish) is shown in figure 8. The interface allows to select a model (ω,δ,μ,γ) for classifying selected tracks from MIDI files. The classification of each extracted window is shown in a row and encoded by colors. Each window content can be played individually and its description visualized.



Clasificador Estilos Musicales MIDI

Midi

Fichero mozart00.mid

Pista Melody (BB)

Modelo

Tipo Clasificador Bayesiano

Ventana 19

Desplazamiento 4

Descriptores Modelo de 12

Información Adicional

Titulo Eine Kleine Nachtmusik, K525, 4th mvt.

Autor Author

Tempo

Duración

Compases

Compas

Reproducción

Resultado Clasificación

Muestra

notas silSIG tesit notMDR DTnota DTdur DTioi itvRNG DTitv DTND sincop itvNOR

Global

Jazz 🟩 29 % Clasica 🟥 71%

Fig. 8. Graphical user interface for automatic music genre recognition.

7. Acknowledgments

This work is supported by the spanish national projects: GV06/166 and CICyT TIN2006–14932–C02, partially supported by EU ERDF.

8. References

M. Baroni (1978). *Proposal for a Grammar of Melody: The Bach Chorales*. Les Presses de l'Université de Montréal.

S. G. Blackburn (2000). *Content Based Retrieval and Navigation of Music Using Melodic Pitch Contours*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, UK.

L. Breiman (2001). Random forests. *Machine Learning*, 45(1):5–32.

G. Buzzanca (2002). A supervised learning approach to musical style recognition. In Music and Artificial Intelligence. Additional Proceedings of the Second International Conference, ICMAI 2002.

W. Chai and B. Vercoe (2001). Folk music classification using hidden Markov models. In *Proc. of the Int. Conf. on Artificial Intelligence*, Las Vegas, USA.

D. Cope (1996) *Experiments in Musical Intelligence.*, volume 2. Cambridge University Press, New York, NY, USA.

P. P. Cruz, E. Vidal, and J. C. Pérez-Cortes (2003) Musical style identification using grammatical inference: The encoding problem. In Alberto Sanfeliu and José Ruiz-Shulcloper, editors, *Proc. of CIARP 2003*, pages 375–382, La Habana, Cuba.

R. Dannenberg, B. Thom, and D. Watson (1997). A machine learning approach to musical style recognition. In *Proceedings of the International Computer Music Conference (ICMC'97)*, pages 344–347.

S. Dubnov and G. Assayag (2002). *Mathematics and Music*, chapter 9, pages 147–158. Springer.

R. O. Duda, P. E. Hart, and D. G. Stork (2000). *Pattern Classification*. John Wiley and Sons.

J. Eggink and G. J. Brown (2004). Extracting melody lines from complex audio. In *Intl. Conf. on Music Information Retrieval*.

A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith (1995). Query by humming: Musical information retrieval in an audio database. In *Proc. of 3rd ACM Int. Conf. Multimedia*, pages 231–236.

A. E. Gomez, A. Klapuri and B.Meudic (2003). Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32-1.

M. Grachten, J. Ll. Arcos, and R. López deMántaras (2004). Melodic similarity: Looking for a good abstraction level. In *Proceedings of the 5th International Conference onMusic Information Retrieval*.

G. R. Hancock and A. J. Klockars (1996). The quest for alpha; developments in multiple comparison procedures in the quarter century since games (1971). *Review of Educational Research*, 66(3):269–306.

I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos (2007). Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithmfor symbolicmusical data. In *Proceedings 4th Sound and Music Computing Conference (SMC'2007)*, Lefkada.

Pl. R. Illescas, D. Rizo, and J. M. Iñesta (2007). Harmonic, melodic, and functional automatic analysis. In *Proceedings of the 2007 International Computer Music Conferrence*, volume I, pages 165–168.

Y. E. Kim, W. Chai, R. Garcia, and B. Vercoe (2000). Analysis of a contour-based representation for melody. In *ISMIR*.

K. Lemstrom and J. Tarhio (2000). Searching monophonic patterns within polyphonic sources. In *Proceedings of the RIAO Conference, volume 2*, pages 1261– 1278.

M. Li and R. Sleep (2004). Melody classification using a similarity metric based on kolmogorov complexity. In *Proc. Sound and Music Computing.*

S. T. Madsen and G. Widmer (2007). Towards a computational model of melody identification in polyphonic music. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 459–464.

A. Marsden (1992). Modelling the perception of musical voices: a case study in rulebased systems. In *Computer Representations and Models in Music*, pages 239–263. Academic Press.

C. McKay and I. Fujinaga (2004). Automatic genre classification using large high-level musical feature sets. In *Int. Conf. on Music Information Retrieval, ISMIR 2004*, pages 525–530.

C. McKay and I. Fujinaga (2006a). jSymbolic: A feature extractor for midi files. In *Proceedings of the International Computer Music Conference*, pages 302–305.

C. McKay and I. Fujinaga (2006b). Musical genre classification: Is it worth pursuing and how can it be improved? In *International Conference on Music Information Retrieval*, pages 101–106.

C.McKay and I. Fujinaga (2007a). Style-independent computer-assisted exploratory analysis of largemusic collections. *Journal of Interdisciplinary Music Studies*, 1(1): 63–85.

C. McKay and I. Fujinaga (2007b). jWebMiner: A web-based feature extractor. In *Proc. of the International Conference on Music Information Retrieval*, pages 113– 114.

E. Narmour (1990). The Analysis and Cognition of Basic Melodic Structures. University Of Chicago Press.

F. Pachet, G. Westermann, and D. Laigre (2001). Musical datamining for emd. In *Proceedings of the Wedelmusic Conference*.

E. Pampalk, S. Dixon, and G. Widmer (2003). Exploring music collections by browsing different views. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR'03)*, pages 201–208, Baltimore, USA.

C. Pérez-Sancho, J.M. Iñesta, and J. Calera-Rubio (2004). Style recognition through statistical event models. In *Proc. of the Sound and Music Computing Conference*.

J. Pickens (2001). A survey of feature selection techniques for music information retrieval. Technical report, Center for Intelligent Information Retrieval, Departament of Computer Science, University of Massachussetts.

P. J. Ponce de León and J. M. Iñesta (2003). Feature-driven recognition of music styles. In *1st Iberian Conference on Pattern Recognition and Image Analysis. LNCS, 2652*, pages 773–781.

P. J. Ponce de León, J. M. Iñesta, and C. Pérez-Sancho (2004). A shallow description framework formusical style recognition. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, 3138.

P. J. Ponce de León, D. Rizo, and J. M. Iñesta (2007). Towards a human-friendly melody characterization by automatically induced rules. In Simon Dixon, David Brainbridge, and Rainer Typke, editors, *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 437–440, Vienna. Austrian Computer Society.

P. J. Ponce de León, D. Rizo, and R. Ramirez (2008). Melody characterization by a fuzzy rule system. In *Proccedings of theMusic and Machine Learning Workshop*.

D. Rizo, J.M. Iñesta, and P.J. Ponce de León (2006a). Tree model of symbolic music for tonality guessing. In *Proc. of the IASTED Int. Conf. on Artificial Intelligence and Applications, AIA 2006*, pages 299–304, Innsbruck, Austria. IASTED, Acta Press.

D. Rizo, K. Lemström, and J.M. Iñesta (2008). Tree structured and combined methods for comparing metered polyphonic music. In *Proc. Computer Music Modeling and Retrieval 2008 (CMMR'08)*, pages 263–278, Copenhagen, Denmark, Copenhagen, Denmark.

D. Rizo, P. J. Ponce de León, C. Pérez-Sancho, A. Pertusa, and J. M. Iñesta (2006b). A pattern recognition approach for melody track selection in midi files. In Tindale A. Dannenberg R., Lemström K., editor, *Proc. of the 7th Int. Symp. on Music Information Retrieval ISMIR 2006*, pages 61–66, Victoria, Canada.

S. Sadie and G. Grove (1984). The New Grove Dictionary of Music and Musicians. Macmillan.

E. Selfridge-Field (1998). *Conceptual and representational issues in melodic comparison*, volume 11 of *Computing in Musicology*, pages 3–64. Cambridge, Massachusetts: MIT Press.

J. F. Serrano and J. M. Iñesta (2006). Music information retrieval through melodic similarity using hanson intervallic analysis. *Research in Computing Science*, 20: 131–142.

H. Soltau, T. Schultz, M. Westphal, and A. Waibel (1998). Recognition of music types. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-1998)*, pages 1137–1140.

C. Spevak, B. Thom, and K. Höthker (2002). Evaluating melodic segmentation. In *ICMAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence*, pages 168–182, London, UK. Springer-Verlag.

E. Stamatatos and G. Widmer (2002). Music performer recognition using an ensemble of simple classifiers. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 335–339.

M. Tang, C. L. Yip, and B. Kao (2000). Selection of melody lines for music databases. In *Proceedings of Annual Int. Computer Software and Applications Conf. COMPSAC*, pages 243–248.

D. Temperley (2004). The Cognition of Basic Musical Structures. The MIT Press.

B. Thom (2000). Unsupervised learning and interactive Jazz/Blues improvisation. In *Proceedings of the AAAI2000*, pages 652–657.

E. Toch (1997). La melodía (translation of 'Melodielehre', 1923). Span Press Universitaria.

P. Toiviainen and T. Eerola (2001). Method for comparative analysis of folk music based on musical feature extraction and neural networks. In *III International Conference on Cognitive Musicology*, pages 41–45.

R. Typke, P. Giannopoulos, R.C. Veltkamp, F. Wiering, and R. van Oostrum (2003). Using transportation distances for measuring melodic similarity. In *Proceedings of the 4th International Conference on Music Information Retrieval.*

G. Tzanetakis, A. Ermolinskyi, and P. Cook (2003). Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143– 152.

A. Uitdenbogerd and J. Zobel (1999). Melodic matching techniques for large music databases. In *Proceedings of the seventh ACM International Multimedia Conference (Part 1)*, pages 57–66. ACMPress.

A. L. Uitdenbogerd and J. Zobel (1998). Manipulation of music for melody matching. In *Proceedings of the sixth ACM International Multimedia Conference*, pages 235– 240. ACMPress.

P. van Kranenburg and E. Backer (2004). Musical style recognition - a quantitative approach. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, pages 106–107.

B. Whitman, G. Flake, and S. Lawrence (2001). Artist detection in music with minnowmatch. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 559–568.

I. H. Witten and E. Frank (1999). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann.

J. Zhu, X. Xue, and H. Lu (2004). Musical genre classification by instrumental features. In *Int. Computer Music Conference, ICMC 2004*, pages 580–583.