Detecting Solo Phrases in Music using Spectral and Pitch-related Descriptors*

Ferdinand Fuhrmann, Perfecto Herrera, and Xavier Serra

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
{ferdinand.fuhrmann, perfecto.herrera, xavier.serra}@upf.edu

Abstract

In this paper we present an algorithm for segmenting musical audio data. Our aim is to identify solo instrument phrases in polyphonic music. We extract relevant features from the audio to be input into our algorithm. A large corpus of audio descriptors was tested for its ability to discriminate between *solo* and *non-solo* sections, which resulted in a subset of five best features. We derived a two-stage algorithm that first creates a set of boundary candidates from local changes of these features and then classifies fixed-length segments according to the desired target classes. The output of the two stages is combined to derive the final segmentation and segment labels. Our system was trained and tested with excerpts from classical pieces and evaluated using full-length recordings, all taken from commercially available audio. We evaluated our algorithm by using precision and recall measurements for the boundary estimation and introduced new evaluation metrics from image processing for the final segmentation. Along with a resulting accuracy of 77%, we demonstrate that the selected features are discriminative for this specific task and achieve reasonable results for the segmentation problem.

1 Introduction

In recent years we have seen a dramatic increase in the size of audio and video databases, creating a growing demand for Music Information Retrieval applications (Orio, 2006). Automatic indexing avoids time consuming meta-data editing and is therefore desirable for managing big digital collections. Determining the presence or absence of a solo phrase in a certain piece of music would help systems in efficiently browsing long archives. Moreover, from an educational point of view, a system able to detect solos in music can help musicians in finding relevant phrases in large digital libraries.

In the literature, solo phrases have been widely used for different research tasks. Many algorithms for musical instrument recognition, melody extraction, pitch detection, and music transcription rely on the fact that there is a predominant instrument in the analysed audio. In the case of large music collections, extracting the relevant data by hand is not feasible and the above mentioned applications would greatly benefit from an automatic segmentation system. Thus, one would have quick access to a huge amount of solo instrument data and model these data for his specific task.

Much research has gone into audio segmentation, most of the effort so far focused on the segmentation of speakers or audiovisual data. These systems estimate the boundaries where there is a change of speaker or the boundaries between rough multimedia categories such as speech, music, etc.in a single audio file. Zhang and Kuo (2001) used a metric-based approach to segment audio data into speech, music, song, environmental sounds etc.: a time series of musical features is obtained from the audio stream and scanned by a sliding window; this window is split in the middle and the difference between a statistic, calculated on both sides, is derived. The peaks of the difference curve are detected and marked as segment boundaries. Cheng and Wang (2004) used this metric-based approach and combined it with a model selection algorithm. The Bayesian Information Criterion (BIC) is used to evaluate the change in a model of segments of the signal. Again, the audio is passed through a window splitting it into two parts. The BIC value, a criterion of how well the data fit a model of a specific complexity (e.g. a certain Gaussian distribution), is calculated for the whole window and its subparts. If the difference of

these values exceeds a certain threshold, a change point is detected. Finally a hierarchical segmentation approach based on classification of the audio data is used by Lu, Jiang, and Zhang (2001): musical features are extracted frame-wise and classified by a simple 2 Nearest Neighbour (NN) algorithm into speech and non-speech classes. The non-speech class is further split into silence, music and environmental sounds using threshold values on the applied musical features.

In contrast to the aforementioned systems, research on segmentation of pure musical signals mainly concentrates on determining the musical structure. Several approaches have been presented using similarity measurements to partition a song into its structural sections (i.e. intro, verse, chorus, etc.). Foote (2000) presented a general system that analyses a novelty score based on a kernel correlation applied to the similarity matrix of a song. This approach was used by Ong and Herrera (2005) for a two stage segmentation process with basic spectral features (MFCCs, subband-energies) to obtain the similarity matrix and a final analysis of the distances of the extracted segments to find large structural blocks. Chroma features are used by Goto (2006) as an input into a similarity measure to detect the chorus sections in a pop song. These descriptors, which characterize the overall timbral and tonal distribution of the analysed audio, are more reliable than spectral features for detecting structural repetitions as they are not as sensitive to timbre variations as MFCCs. A rather new approach for detecting these structural boundaries in music was presented by Turnbull, Lanckriet, Pampalk, and Goto (2007). Here a supervised classification of boundary and non-boundary segments was compared to a simple peak-picking algorithm based on time series of difference features. In their supervised approach the authors derived their features by smoothing the raw feature values with different smoothing lengths and calculating the instantaneous derivatives. The frames are then labelled as "boundary" (if they contain a transition from one section to another) or "non-boundary" (if there is no transition). Finally a decision stump classifier, which generates a one-level decision tree with a threshold value for the feature, is used in combination with a boosting algorithm to weight and combine the individual decision stumps. The classifier was trained with labelled data, and an unknown target song was then segmented according to the output of the boosted decision stump.

The specific task of detecting solo instrument phrases within musical audio data has not been addressed much by researchers in the past. Peterschmitt, Herrera, and Gómez (2001) tried to locate solo phrases by a confidence measure of a monophonic pitch detector. The Two Way Mismatch (TWM) (Maher & Beauchamp, 1994) algorithm was trained on a specific instrument and applied to the instruments' solos. Although the initial observations were promising, the overall results did not satisfy the prospects of the research: the derived decision function was far too noisy to discriminate between solo and non solo parts and resulted in a percentage of 56% correctly assigned frames. Pitch detection via autocorrelation together with a cancellation filter was used by Smit and Ellis (2007) in order to find single voice phrases in baroque music. The output of the autocorrelation was fed into a comb filter to cancel all harmonic parts of the analysed signal excerpt. The error output of the filter was then classified using a simple Bayesian classifier to obtain the likelihoods, and a Hidden Markov Model (HMM) was applied to find the best label sequence. Finally, Eggink and Brown (2004) used the probabilistic output of an instrument recognition module to indicate the presence of the solo instrument. Reported results included an accuracy of 65% correctly assigned frame states.

In this paper we present a system able to detect solo phrases in music that is based on spectral and pitch related analysis of the audio. In the context of this work, as it concentrates on classical music, we define as *solo* a musical passage, where one instrument is accompanied by a background orchestra or is playing completely alone. This implies that the soloing instrument is predominant among the instruments present. Moreover, the soloist is playing in a rather free manner by not exactly repeating longer formal or melodic structures. For the computational modelling we extract features from the audio for estimating boundary candidates and combining them with the predictions of a trained classifier. The classifier assigns the target labels *solo* and *non-solo* to fixed-length segments of audio. In section 2 we describe the feature selection, the boundary estimation and the label assignment process of the presented algorithm. Section 3 covers details of the used data and the experimental setup. We introduce new segmentation evaluation metrics and present the final results. In the subsequent discussion we analyse these results which substantiate our approach. Finally section 5 contains conclusions and indication of future research work.

2 Description of the proposed algorithm

In this section we present details of our algorithm for locating solo phrases inside a given audio file. An example of solo phrases is given in Figure 1. Here, the spectral representations of one audio frame are shown for two solo examples. The first one illustrates a solo, where one instrument is playing alone. Thus, no accompaniment is present and the harmonic structure of the instrument sound can be clearly seen. The bottom example is taken from the same piece of music with the same instrument soloing but now with orchestral accompaniment. The

harmonic structure can still be seen but the spectrum of the overall sound is also influenced by the components of the accompanying instruments. Our system tries to capture both kinds of solos shown in Figure 1, as they behave

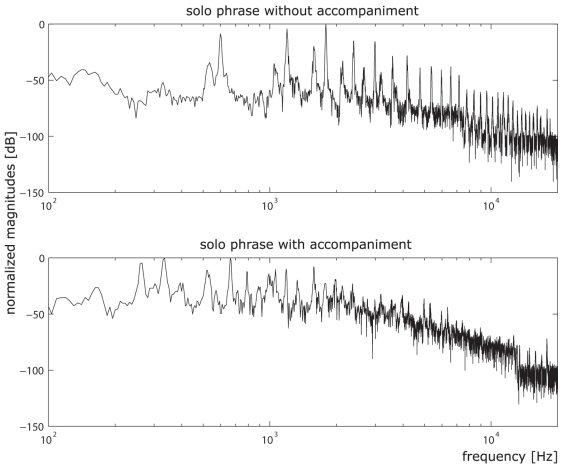

Figure 1: Two examples of a solo. Spectral representation of one audio frame.

similar with regard to their spectral properties. If the solo is accompanied, its recognition depends largely on how much the accompaniment contributes to the overall sound.

The next subsections cover the structural blocks of the proposed algorithm. Figure 2 shows an overview of the system.

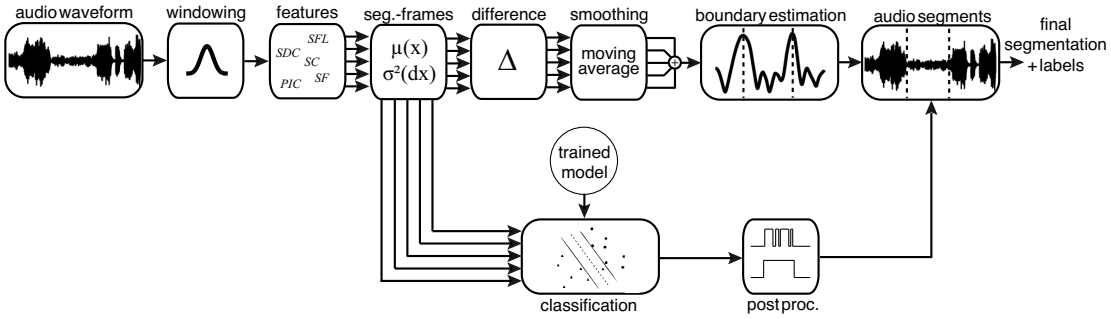

Figure 2: Graphical illustration of the proposed algorithm. Audio features are computed from signal frames and grouped to analysis segments, which are input into boundary estimation and classification. The predictions of the classifier are used to assign labels to the regions derived by the boundaries, yielding the final segmentation.

2.1 Musical features

For determining the best set of descriptors able to distinguish the two target classes, we performed feature selection on our dataset. A large set of features was tested in the experiments described in section 3.3. We ended up with a small subset consisting of five descriptors, all highly discriminative for solo and non-solo phrases.
Features chosen for the segmentation algorithm are:

- **Pitch Instantaneous Confidence (PIC):**

  This feature is derived from the pitch estimator proposed by Brossier (2006). Its value determines the depth of the deepest valley of the yin_{fft}-lag function[1]. This function represents the periodicities inside an audio excerpt. The descriptor is computed as follows:

$$PIC_n = 1 - min(yin^n_{fft}(\tau)), \tag{1}$$

[1]The feature is an enhancement of the YIN algorithm presented by Cheveigne and Kawahara (2002). Its name is derived from the well-known term from Chinese philosophy.

$$yin^n_{fft}(\tau) = \frac{4}{N}\sum_k |X_n(k)|^2 - \frac{2}{N}\sum_k |X_n(k)|^2 \cos(\frac{2\pi k\tau}{N}), \quad (2)$$

where N is the frame size, k the frequency bin index, τ the time lag in samples, X the magnitude spectrum weighted by the outer ear transfer function (Terhardt, 1979), and n the frame index. The feature is normalised between 0 and 1 and indicates the strength of the predominant pitch in the analysed audio. The higher the value, the more predominant the estimated pitch, resulting in a value of 1 for a monophonic, single pitched sample. Solo sections therefore produce a higher pitch confidence than non-solo sections, as they usually contain a predominant instrument.

- **Spectral Dissonance (SDC):**

  This is a perceptual feature that is used to obtain a roughness measure of the analysed sound. It is based on the fact that two close-in-frequency spectral components form an interference pattern that perceptually yields dissonance sensations (Plomp & Levelt, 1965). These can be characterised by the frequency and amplitude relations of the components. The overall dissonance is then calculated by summing up the dissonance values of all the component pairs of a given audio frame. In the case of polyphonic music the number of components is influenced by the number of concurrent instruments, as each instrument contributes a set of harmonics to the audio. Given the values of all pairs of components (i.e. the spectral peaks) of a certain spectrum, the dissonance is calculated as follows:

$$SDC_n = \sum_{i,j}(1 - c(i,j))\overline{X}_n(j), \quad (3)$$

  where the function c represents a polynomial implementation of the consonance curves obtained by Plomp and Levelt (1965) using human ratings. $\overline{X}_n(j)$ is the normalised amplitude of the frequency component at bin j, weighted with the outer ear frequency response (Terhardt, 1979) at frame n. As most of the spectral components of ensemble section carry a similar amount of energy and their frequency locations slightly differ from perfect integer ratios (i.e. "perfect tuning"), their individual dissonances result in higher values. This affects the sum in Eq. 3 and produces higher overall values compared to a solo section, where the ratio between the energies of the components of the solo instrument and the accompaniment is significantly different and therefore the above mentioned mistuning has less impact.

- **Spectral Flux (SFL):**

  Spectral Flux measures the change of the spectrum, derived by comparing the spectrum of the current frame to the spectrum of the previous one. More precisely, it takes the Euclidean norm between the two spectra, each one normalized by its energy.

$$SFL_n = ||\overline{X}_n(k) - \overline{X}_{n-1}(k)||_2, \quad (4)$$

  where $\overline{X}_n$ denotes the normalized magnitude spectrum at frame n. Spectral Flux values of neighbouring frames in solo section show small values, as the respective spectra are mainly influenced by only one instrument. The opposite is true for ensemble sections, where multiple instruments contribute to the mixture.

- **Spectral Flatness (SF):**

  A well known feature from the MPEG-7 standard (Salembier & Sikora, 2002). It is defined by the relation of the geometric to the arithmetic mean of the spectrum. Its output can be seen as a measure of the tonality/noisiness of the sound. A high value indicates a flat spectrum, typical of noise-like sounds, or, in our case, ensemble sections. On the other hand, harmonic sounds produce low flatness values, an indicator for solo phrases. Izmirili (2000) proposed first to use it to locate the onsets of solo phrases in music.

$$SF_n = ((\prod_k X_n(k))^{1/K})(\frac{1}{K}\sum_k X_n(k))^{-1}, \quad (5)$$

  where k is the frequency bin index of the magnitude spectrum X at frame n.

- **Spectral Crest (SC):**

  This feature describes the shape of the spectrum. High crest values indicate a peaky spectrum, noise like spectra will have a value near 1. For solos, where the spectrum is dominated by the harmonic components of a single instrument, the spectral crest will have higher values than for phrases with no soloing instrument. It is calculated by

$$SC_n = (\max(X_n(k)))(\frac{1}{K}\sum_k X_n(k))^{-1}, \quad (6)$$

where X denotes the magnitude spectrum at frame n.

Figure 3 shows the average evolution of the standardised (i.e. zero mean and unit variance) features of 38 different solo excerpts at the intersection of a solo phrase. The time axis is normalized to the duration of the segments, the vertical line indicates the manually annotated boundary.

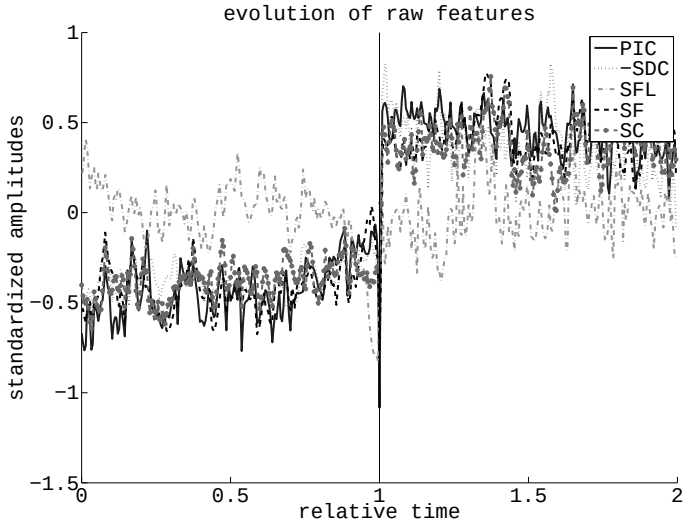

Figure 3: Average evolution of the selected features for 38 excerpts containing a solo phrase. It covers the transition from a non-solo to a solo segment; the black vertical line indicates the class boundary according to the manual expert-based annotations.

Regarding the individual importance of the five selected features in the subsequent boundary estimation and classification (note that the individual feature amplitudes in Figure 3 are standardised), we observed that the *SDC* feature is the most important descriptor after which the remaining descriptors imply an equal amount of predictability. By building a model with a C4.5 decision tree algorithm on our training data (see Section 3.1 for details), the SDC represented the top-level feature of the developed tree, indicating its ability to linearly separate the two classes. Moreover, applying a logistic regression on the same training data set, the obtained attribute weights were 2.26, 0.63, 0.56, 0.41, and 0.23 for the *SDC*, *SFL*, *SC*, *SF*, and *PIC*, respectively.

In short, these five features describe the audio under investigation in terms of predominance of a pitched sound. The more predominant a sound is inside the mixture, the higher the pitch confidence and crest, and the lower the flux, flatness and dissonance values. In other words we model our data with a measurement of "salient harmonics". The more salient the spectral components of a given instrument are in the musical context, the more the entire sound is affected by them and the more the selected features will tend to indicate the class *solo*.

2.2 Boundary estimation

In our presented system we use local changes of the features for estimating boundary candidates. We pick peaks from a smoothed difference function derived from the extracted features to find these changes. All peaks above a dynamic threshold are then used as boundary candidates.

First, we extract the features from the windowed audio data (see details in section 3.2). To obtain the difference function, we use a sliding window and calculate the difference between the statistics of the first and the second half (Zhang & Kuo, 2001). Following the results obtained by the feature selection procedure described in section 3.3, we use the mean for the PIC, SF and SC features and the variance of the difference for the SDC and SFL features. As the obtained data are very noisy, we apply a low pass filter to get rid of the feature fluctuations. Therefore we use a moving average filter with a window length of five seconds to smooth the time series. We then standardise every feature by giving it zero mean and unit variance. This is necessary in order to compare the feature values of different musical pieces in our collection. Finally we combine the generated difference features to obtain a one dimensional time series. Since this combination is a crucial step and there is no common procedure for doing it, we tried two different approaches. The first one was simply calculating the Euclidean norm, the second one used

a weighted sum, applying the weights generated by a logistic regression, for every sample. Finally we ended up using the Euclidean norm, giving us slightly better results than the weighted sum.

For every sample in the time series of smoothed differentiated features a dynamic threshold is calculated by averaging a predefined number of preceding samples. A median filter is used to obtain the threshold value for every frame. The peaks are finally found by calculating the difference of the actual feature value to the threshold value. The regions, wherein this difference exceeds a certain value are kept and the maximum inside each region is marked. We then apply the local-peak-local-neighbourhood (LPLN) approach (Turnbull et al., 2007) by looking in the local neighbourhood of these marks in the unsmoothed data to estimate the final peak candidates. This gave us better results than simply picking the peaks from the smoothed data as reported by Turnbull et al. (2007). Figure 4 provides the main steps to derive the boundary candidates from the smoothed difference function.

```
peakfct = smoothed difference function
avLen = length of threshold
startIndex = definded by smoothing, differentiating,
             and threshold length

for frameIndex in numFrames
     if frameIndex>startIndex
          thresh = median(peakfct(frameIndex-
                   avLen:frameIndex-1))
          if peakfct(frameIndex)>level*thresh
               bumps(frameIndex) = peakfct(frameIndex)-tresh
          else
               bumps = 0

find bumps>0

for bump in bumps
     maxbump = frame of maxvalue of bump
     peak = frame of maxvalue in unsmoothed peakfct
            inside tolerance window centered at maxbump
     boundary(peak) = 1
```

Figure 4: Boundary candidate estimation procedure.

We finally use a very low threshold value for determining the boundary candidates in order to get the maximum number of correct boundaries (i.e. a high recall value). Consequently, we have to accept a high number of false positives (i.e. a low value of precision), which results in an oversegmentation of the audio. Since we further classify the content inside adjacent boundaries, we cope with this problem by merging neighbouring segments of the same class after the label assignment.

2.3 Label assignment

After deriving a set of candidates from the peak picking algorithm we assign a label (i.e. *solo* or *non-solo*) to the content between each pair of adjacent boundaries. Therefore we allocate a state to every frame (i.e. 1 for *solo* and 0 for *non-solo*) and use these values for deriving the label for the whole segment.

First we group the frame-based features to segments of five second length and calculate the appropriate statistics, using the same hop size as before (for the remainder of the paper we will use the term segment-frame to refer to these fixed-length segments, so as not to confuse them with neither the raw feature frames nor the segments output by the algorithm). Due to the results of the feature selection described in section 3.3, we use the mean for the PIC, SF and SC features, whereas for the SDC and SFL features we use the variance of the difference. These segment-frames are then classified according to the binary classification task. We generated a classification model using our training data and tested it with the test collection (see section 3.1 for more details on the used data). Once we obtain the class assignment for the segment-frames from the classifier a final postprocessing stage smoothes the data. We use morphological filters, previously used in musical structure analysis (Lu, Wang, & Zhang, 2004) and image processing (Castleman, 1996), to remove segments, which are too short, and to promote longer ones.

To assign the labels to the content between a pair of neighbouring boundary candidates we use a majority vote among the covered segment-frames and their class assignment. We do this for every pair of adjacent boundaries and merge all neighbouring segments of the same class to obtain the final segmentation. Figure 5 shows the principles of the basic label assignment procedure.

```
boundary_pairs = list of all pairs of adjacent boundaries
predictions = segment-frame based classifier output...
              (1 for solo and 0 for non-solo)

counter = 0
for pairs in boundary_pairs:
     counter += 1
     frames = predictions(pairs(1):pairs(2)-1)
     if sum(frames)≥length(frames)/2:
          label(counter) = 'solo'
     else
          label(counter) = 'non-solo'
```

Figure 5: Pseudo code for the label assignment process.

3 Evaluation

3.1 Musical data

For training and testing our algorithm we use a dataset consisting of various classical recordings, each of which is composed for a solo instrument with accompaniment. Six different instruments are covered, namely Cello, Clarinet, Oboe, Piano, Saxophone and Violin. We chose these instruments as they are the most used solo instruments in classical music and a good selection of string and wind instruments. In this database we include pieces of musical periods ranging from baroque music of J. S. Bach to modern concerts for saxophone and orchestra, even including drums. This also implies that we cover a wide range of different instrumentations and playing styles. The collection consists of continuous 30 second excerpts, covering 15 seconds of solo and 15 seconds of ensemble music. A total number of 40 fragments were extracted per instrument, each representing a different piece of music. All together we therefore collected samples from 240 different pieces of classical music. In order to obtain a separated training and testing set, we split the collection assigning half of the tracks of each instrument to each of the two sets.

For the evaluation dataset three whole classical pieces per instrument are used, with a duration between two to 15 minutes. Additionally, we added three pieces of two more instruments (Flute and Horn) to this set to ensure that our algorithm is not trained just for the six instruments from the training and testing sets. Hence, our evaluation set contains 24 different pieces. All evaluation pieces were annotated by a human expert (i.e. the researcher) with the target classes. A list with the titles of all the evaluation tracks used is available online[2].

All files for training, testing and evaluation were taken from commercially available audio CDs with a sampling frequency of 44100 Hz and an amplitude quantisation of 16 bits. The files were converted to MPEG-layer-3 format using an encoding of 192 kbps and finally to mono for further audio processing.

3.2 Experimental setup

In our experiments we create frames using a Blackmann-Harris 62dB window of 185 ms length, with a hop size of 93 ms. We use a 8192-point FFT to extract our features.

3.3 Feature selection

Since the detection of solo instrument parts in music has not been addressed much in literature so far, there exist, to our knowledge, no studies dealing with audio features able to discriminate between solo and non-solo phrases. We therefore conducted a study to determine the best set of descriptors for this task.

For these experiments we used our collected data from the training database, including the instruments Cello, Clarinet, Oboe, Piano, Saxophone and Violin. As we wanted to capture the properties of our features at the transition of a solo to a non-solo phrase (and vice versa), we first analysed the evolution of a large set of audio features in sections where the music changes from a solo to a non-solo phrase (and the other way round). A complete list of tested features can be found in Table 1.

Audio excerpts of ten seconds length were extracted from the musical data, covering the transition from solo to non-solo (and vice versa), and divided at the expert annotated boundary. Additionally, to validate the results of the first step and avoid potential errors in the feature value caused by the phrase transition, we built ten *random* datasets from the training database. One solo and one ensemble segment with a duration of five seconds were

[2] http://mtg.upf.edu/files/personal/evaluationfiles.pdf

Table 1: A list of all audio features tested in this work.

spectral centroid	energyband low
spectral spread	energyband middle low
spectral skewness	energyband middle high
spectral kurtosis	energyband high
barkbands spread	spectral flux
barkbands skewness	spectral hfc
barkbands kurtosis	spectral rolloff
pitch instantaneous confidence	spectral strongpeak
pitch salience	spectral dissonance
spectral energy	spectral flatness
rms	spectral decrease
zerocrossingrate	spectral crest
spectral complexity	

randomly selected and concatenated from every audio file of the training data set, repeating this for each of the ten datasets.

From these data the audio features were extracted on a frame basis. Several statistics, consisting of the mean, variance, mean of difference and variance of difference, were computed for every feature over the length of the annotated segments within the generated excerpt. The change in these values from the solo to the non-solo phrase should indicate the ability of the feature to discriminate between the two classes. The data were then labelled according to their classes (i.e. solo and non-solo) and input into the WEKA machine learning software (Witten, Franck, Trigg, Hall, & Holmes, 1999). We used the correlation based feature selection algorithm (Hall, 2000) to find the best set of descriptors able to classify the data according to its given labels. This method searches the feature space for subsets of descriptors, taking the correlation of the features with the class and the intercorrelation of the features inside the subset into account. More precisely, the goodness of a feature subset S containing k features is defined as follows:

$$Merit_S = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}}, \tag{7}$$

where $\overline{r_{cf}}$ denotes the average feature-class correlation and $\overline{r_{ff}}$ the average feature-feature intercorrelation. If the problem at hand is classification (i.e. with discrete class data), the numerical input variables in Eq. 7 must be discretised and the degree of association between different variables is computed by symmetrical uncertainty (Press, Teukolsky, Vetterling, & Flannery, 1992)

$$U(X,Y) = 2 \times [\frac{H(Y)+H(X)-H(X,Y)}{H(X)+H(Y)}], \tag{8}$$

where $H(X)$ denotes the entropy of X.

To derive the feature subsets in a reasonable amount of computational time (in general, evaluating all 2^n possible feature subsets is not feasible, with n being the total number of features), the method utilises the *best first* search algorithm (Hall, 2000) to limit the subsets of features to evaluate.

For every dataset, a list of the most discriminative features was generated, altogether eleven lists (one list for the *transition* and ten for the *random* datasets). We then collapsed the feature lists of the *random* databases by taking only the features which appear in at least eight of the lists. The final subset of features was found by comparing the *transition* to the resulting *random* feature list and taking the descriptors which appear in both lists. This resulted in the features PIC, SF and SC (mean value along the segment-frame) and the SDC and SFL features (variance of difference along the segment-frame). For details see section 2.1. With this subset we evaluated the performance of several classifiers in WEKA to classify the generated segment-frames into solo and non-solo.

Figure 6 shows box-plots of the selected features, calculated from the data of the training set, with respect to the classes, i.e. *solo* and *non-solo*. The features were standardised, grouped into 5-second segment-frames, and the appropriate statistic was calculated therein. To guarantee as much statistical independence as possible, adjacent segment-frames did not overlap, which resulted in approximately 730 instances. Additionally, Table 2 shows z-values at a significance level of 95%. These are obtained performing a non-parametric Mann-Whitney-U-test, which is used to determine statistically significant differences in the population means of two distributions (Hollander & Wolfe, 1999). The significance of the resulting value is then evaluated in a table of the Normal (z) distribution. The same data as for the box-plots was used for this test.

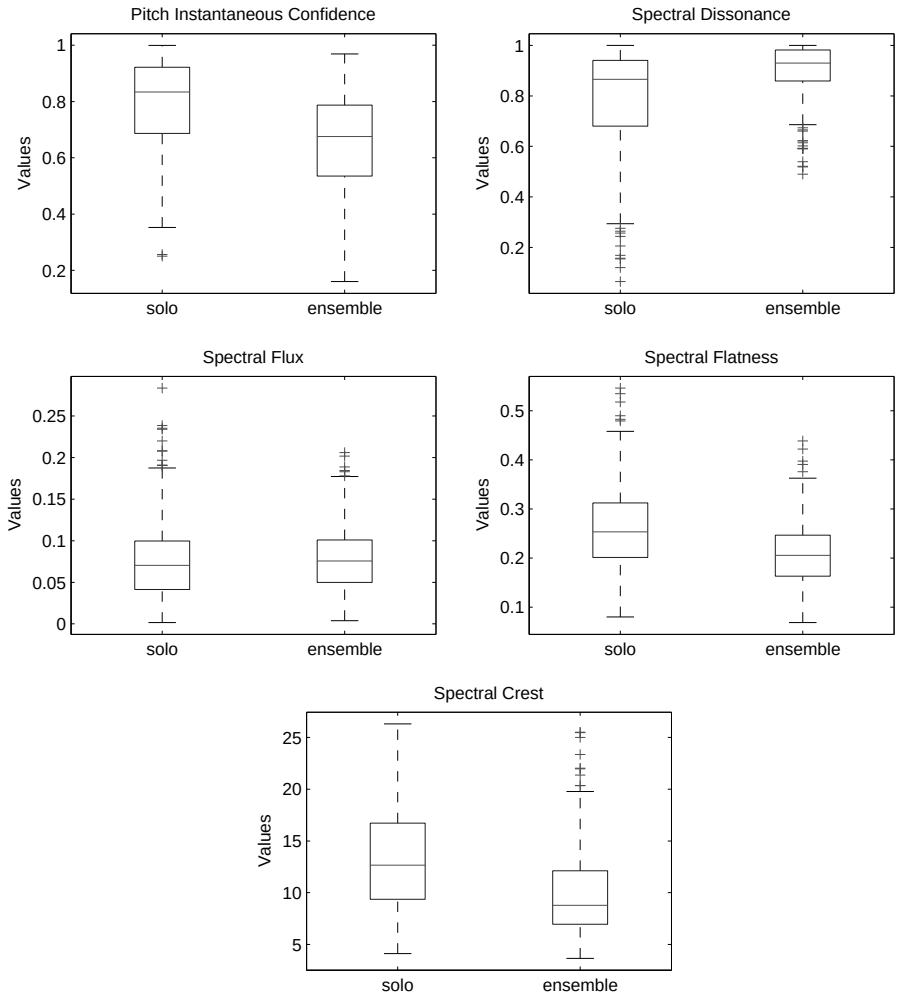

Figure 6: Box-plots of the selected features computed for the training data set.

Table 2: *z*-values of the selected features as a result of the Mann-Whitney-U-Test, performed on the training data set with respect to the classes. The double asterisk indicates statistical significance with a *p*-value smaller 10^{-3}.

feature	*z*-value
PIC**	−10.87
SDC**	8
SFL	1.16
SF**	−9.22
SC**	−9.57

Results of the test indicate statistical significance for all features except the SFL (see also the respective box-plot in Fig. 6). However, the significance test does not account for informative feature combinations, as it evaluates the distribution of the features separately. We kept the feature for the experiments in the subsequent sections – motivated by the subset evaluation procedure included in the feature selection process (i.e. the evaluation of the predictiveness of a set of features) – as it might carry some complementary information, thus important for the

recognition performance.

3.4 Classifier performance

In order to derive the segment-frame based class assignment we studied the performance of several classifiers. Different classification models were trained and tested, using the data of the training and testing set. From these data we extracted the selected features, grouped them to segment-frames of five seconds length using a sliding window, and calculated the appropriate statistics. The generated feature vectors were labelled according to their class and input into WEKA for testing six different classifiers. We obtained similar classification results for the artificial neural network, the C4.5 decision tree, the logistic classifier and the two support vector machines SMO (sequential minimal optimization) and libSVM (Chang & Lin, 2001) (both with a RBF kernel), using the default parameter settings for every classifier. Table 3 shows the results for the six different classifiers, trained on the excerpts of the training set and tested with the independent test set.

Table 3: Accuracy of classifiers trained with the training and tested with the testing dataset, using default parameter settings.

10NN	C4.5	ANN	logistic	SMO	libSVM
79.9%	82.3%	82.4%	82.5%	82.7%	82.7%

Given the obtained results listed in table 3, four of the five tested algorithms (we treat libSVM and SMO as the same algorithm) perform more or less the same on the given data. However, we finally decided to use the support vector machine, as performance improved above two percent points after parameter tuning (we performed a grid search over the complexity parameter of the classifier and the gamma parameter of the radial basis function (RBF) kernel). Furthermore, the libSVM clearly outperformed the SMO algorithm regarding computation time, being about five times faster. Moreover, its functionality is intuitive and the models were easier to implement in our developing environment.

3.5 Parameter estimation for the boundary detection

First we fixed the length of the difference window to ten seconds, to be consistent with our feature selection experiments described above. Then we evaluated the values for the smoothing and threshold-calculation length in a certain parameter range. The final values, five seconds for both smoothing and threshold length, represent a good trade-off between acceptable temporal resolution for peak picking and a good reduction of the fluctuations in the difference-feature time series.

3.6 Evaluation metrics

In our experiments we evaluated the boundary estimation and the classifier performance, separately and combined. We compare the boundary candidate estimation to the boundaries of the final segmentation, and the classifier performance to the overall performance of the final segmentation. Precision and recall measurements are used for the boundary evaluation, we consider a boundary as correctly detected if the absolute time difference between the estimate and the true boundary is smaller than two seconds. This value, which corresponds to less than one bar for a piece of music in quadruple meter at 90 beats per minute (bpm), was set ad hoc and is an affordable error in many music audio description systems. For the evaluation of the classification and the overall segmentation we first calculate the percentage of correctly assigned frames for each class (i.e. *solo* and *non-solo*) by dividing the number of correctly assigned frames by the overall number of frames of one class and multiplying it by 100. We will use the terms S_{corr} and NS_{corr} for the solo and non-solo case, respectively. Second, we calculate the percentage of correct decisions *CD*, which is a quantitative evaluation of the overall correct assigned frames. It is calculated by

$$CD = (1 - P(error)) \times 100, \tag{9}$$

with

$$P(error) = P(Solo)P(NonSolo|Solo) + P(NonSolo)P(Solo|NonSolo), \tag{10}$$

where $P(NonSolo|Solo)$ is the fraction of *Solo* frames classified as *NonSolo* (i.e. the false negative rate, when regarding a *solo* vote as a positive) and $P(Solo)$ is the expected value of the class *Solo*, which refers to the amount

of *solo* frames in the ground-truth with respect to the total number of frames (Yang, Albregtsen, Lonnestad, & Grottum, 1995).

Furthermore we introduce new performance measurements taken from image segmentation (Ortiz & Oliver, 2006). These evaluation metrics have not, to our knowledge, been used in Music Information Retrieval (MIR) so far. In contrast to the above mentioned metrics they capture the quality of the output segmentation with respect to the annotated ground truth by evaluating the intersections of the output and the reference segments. We adapt metrics presented by Ortiz and Oliver (2006), which take the correct grouping of frames and under/oversegmentation into account. Here, undersegmentation refers to the case where a single output segment covers several segments of the expert-based manual annotation. Accordingly, oversegmentation appears if one ground-truth segment is split into several output segments. For qualitatively capturing these effects we first construct the overlapping area matrix (OAM) (Beauchemin & Thomson, 1997) with the output of the algorithm and the ground-truth segmentation. Every entry $C_{i,j}$ of this matrix contains the number of frames in the overlap between the output segment j and the reference segment i. For perfect segmentation (i.e. same number of segments in reference and output segmentation and no over/undersegmentation) the OAM contains non-null entries only on its diagonal, each entry representing the number of frames of the corresponding segment. In the case of segmentation errors we will find off-diagonal non-null entries, which characterise the error due to over/undersegmentation. In what follows, $\sum_j C_{i,j}$ is the number of frames in the ground-truth segment i and $\sum_i C_{i,j}$ is the number of frames in the output segment j. From the OAM matrix we derive three evaluation metrics (Ortiz & Oliver, 2006):

- Percentage of correctly grouped frames:

$$CG(p) = \frac{1}{n_t} \sum_{i=1}^{N_r} \sum_{j=1}^{N_o} (CR(S_{ref,i}, S_{out,j}, p) \times C_{i,j}) \times 100 \tag{11}$$

with

$$CR(S_{ref,i}, S_{out,j}, p) = \begin{cases} 1 & \text{if } \frac{C_{i,j}}{n(S_{out,j})} \geq p, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

and

$$n(S_{out,j}) = \sum_{i=1}^{N_r} C_{i,j}, \tag{13}$$

where N_r and N_o are the number of segments in the ground-truth and output segmentation, respectively, and n_t the total number of frames in the target audio. $S_{ref,i}$ refers to the reference segment i, $S_{out,j}$ to the output segment j and p is the penalty factor.

The *Percentage of correctly grouped frames CG* accounts for those frames in a ground-truth segment $S_{ref,i}$, which are concentrated in a single output segment $S_{out,j}$. It describes to what degree the segmentation algorithm does not group frames from multiple reference segments into a single output segment. For perfect segmentation its value is 100% and any single frame error would reduce it dramatically. Therefore the penalty factor p is introduced to relax the constraint of perfect segmentation to *nearly* perfect segmentation, where the term *nearly* depends on the value of p. This parameter describes the amount of tolerance used for the measurements (a value of 1 indicates the most restrictive scenario).

- Percentage of Undersegmentation:

$$US(p) = \frac{1}{n_t} \sum_{j=1}^{N_o} ((1 - UR(S_{out,j}, p)) \times n(S_{out,j})) \times 100 \tag{14}$$

with

$$UR(S_{out,j}, p) = \begin{cases} 1 & \text{if } \frac{\max_{k=1,\ldots,N_r}(C_{k,j})}{n(S_{out,j})} \geq p, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The *Percentage of Undersegmentation US* represents the amount of frames belonging to a single output segment $S_{out,j}$ which covers several segments of the ground-truth $S_{ref,i}$. As before, the penalty factor p is introduced to tolerate slight errors of the output segmentation. When looking at the OAM matrix the function $UR(S_{out,j}, p)$ works over the columns of the matrix, taking into account those output segments $S_{out,j}$ whose overlap with at least one reference segment $S_{ref,i}$ is greater than or equal to $p \times 100\%$.

- Percentage of Oversegmentation:

$$OS(p) = \frac{1}{n_t} \sum_{i=1}^{N_r} ((1 - OR(S_{ref,i}, p)) \times n(S_{ref,i})) \times 100 \tag{16}$$

with

$$OR(S_{ref,i}, p) = \begin{cases} 1 & \text{if } \frac{\max_{k=1,\ldots,N_O}(C_{i,k})}{n(S_{ref,i})} \geq p, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

and

$$n(S_{ref,i}) = \sum_{j=1}^{N_o} C_{i,j}. \tag{18}$$

The *Percentage of Oversegmentation OS* refers to those output segments $S_{out,j}$, which split a single ground-truth segment $S_{ref,i}$. The function $OR(S_{ref,i}, p)$ works over the rows of the OAM, accounting for those rows, represented by the reference segments $S_{ref,i}$, which have more than one non-null entry. These indicate the splits caused by the corresponding output segments $S_{out,j}$. Again, the penalty factor p is introduced, tolerating a certain amount of segmentation error.

As these evaluation metrics based on the OAM take the overlapping areas of the output and reference segmentation into account, they are able to capture the quality of the segmentation. Unlike many other evaluation metrics, they act on a segment basis and not directly on a frame basis. First, the erroneous output segments are marked with respect to the specific criteria (correct grouping, under- or oversegmentation). Second, all frames of the erroneous segments are accumulated and related to the total amount of frames in the analysed audio. Thus, the amount of error the segment contibutes to the metric depends on its size. Furthermore, the incorporation of the penalty factor p allows them to disregard small segmentation errors, which are very common with this kind of problem. We used a penalty factor of 0.8 in all our subsequent evaluation experiments. Again, its value was set ad hoc, mostly to relax constraints in the evaluation metrics and maximise the meaningfulness of the derived segments.

3.7 Results

Overall performance

Here we show the results for the evaluation dataset. Using libSVM with a radial basis function kernel as the classifier we present the final results for our system in table 4. As mentioned in section 2, we obtain a very low

Table 4: Performance of the different systems on the evaluation dataset. The left column represents only the boundary candidate estimation, the middle one the performance of the segment-frame-based classification and finally the right one the performance of the joint system (combined boundary and classification).

evaluation metric	boundary	classifier	joint system
Precision	0.21	x	0.37
Recall	0.74	x	0.59
S_{corr}	x	74%	74%
NS_{corr}	x	83%	85%
CD	x	76%	77%
$CG(p)$	x	84%	76%
$US(p)$	x	15%	22%
$OS(p)$	x	65%	52%

precision value of 0.21 for the boundary candidate estimates. This is an artifact of the low threshold value we used to derive the peaks from the difference function. However, using this low decision criterion also results in a high value for the boundary recall. Since we merge adjacent segments of the same label after the label assignment process, we are far more concerned about the boundary recall than the precision. The low precision value only implies an over-segmentation of the audio (i.e. a lot of redundant candidates), which is alleviated by the merging process of labelled segments.

Instrument specific performance

Here we compare the performance of the complete system when applied to just one specific instrument of our

evaluation dataset. Table 5 lists the results for the six instruments plus the two additional ones (marked with an asterisk). The performance of the two additional instruments Flute and Horn in Table 5 is comparable to the results of the other six instruments and supports the instrument-independence of our algorithm. The slightly better performance of these additional instruments can be explained by their special sound producing mechanisms, as they produce a well defined harmonic tone with a strong onset. Furthermore, as we only evaluate three pieces per instruments, the boundary results can be slightly biased by an easy-to-segment piece.

Table 5: Instrument specific performance of the presented algorithm (joint system). The additional instruments, which were not used for the training and testing of the algorithm, are marked with an asterisk.

evaluation metric	Cello	Clarinet	Oboe	Piano	Saxophone	Violin	Flute*	Horn*
Precision	0.19	0.34	0.26	0.41	0.35	0.42	0.38	0.58
Recall	0.38	0.66	0.55	0.39	0.61	0.62	0.73	0.77
S_{corr}	71%	72%	73%	74%	76%	72%	76%	78%
NS_{corr}	71%	91%	90%	71%	94%	74%	92%	95%
CD	70%	79%	79%	72%	81%	74%	81%	83%
$CG(p)$	77%	85%	83%	44%	85%	67%	85%	84%
$US(p)$	21%	14%	14%	54%	13%	29%	14%	13%
$OS(p)$	65%	65%	62%	37%	53%	48%	57%	30%

Performance depending on musical period

In this section we grouped our evaluation dataset according to the musical period in which they were composed. From the initial 24 tracks of this dataset we chose 20 tracks to have five pieces in every period. Table 6 shows the performance of the complete system for the four covered musical periods, namely baroque, classic, romantic and contemporary. Even if the dataset is of moderate size, we can see that for periods where the size of the orchestra

Table 6: Comparison of the performance of the overall system on the different musical periods covered by the classical evaluation dataset.

evaluation metric	baroque	classic	romantic	contemporary
Precision	0.33	0.33	0.42	0.34
Recall	0.68	0.53	0.54	0.55
S_{corr}	76%	75%	75%	70%
NS_{corr}	93%	79%	79%	79%
CD	82%	76%	76%	73%
$CG(p)$	87%	72%	68%	70%
$US(p)$	11%	26%	29%	28%
$OS(p)$	50%	61%	49%	49%

is relatively small and the music is bounded by more strict rules (e.g. the baroque period) we achieve better results than in other periods. It is clear that with increasing size of the orchestra the performance of the algorithm decreases (from baroque to wiener classic and further from romantic to modern), as there are more instruments in the mixture and therefore more possible error sources.

Figure 7 illustrates the segmentation of a classical piano concerto visualized with our developed user interface. The vertical lines indicate the ground truth annotations and the rectangles and lower solo bars represent the estimated locations of the solos. In the figure we can observe several examples of incorrect segmentation. A typical error is that the system does not segment very short sections ranging from one to five seconds. Using segment-frames of five seconds length (see details in section 2) for both boundary estimation and classification can blur sections which are entirely covered by one grouping window. This also causes errors when the solo instrument is not playing continuously. In some sections the solo instrument is answered by an ensemble section and vice versa. If the length of these short sections falls below a critical length (i.e. solo and ensemble are covered by one single grouping window), the system often yields an erroneous output. Another error source are “duet” sections where the solo instrument is accompanied by a second instrument with similar loudness. Here the features indicate an ensemble section as there is no predominating instrument. Other errors occur with low-frequency sounds and

crescendos of the accompaniment, as well as with ensemble passages where several instruments of the same type play a main melody.

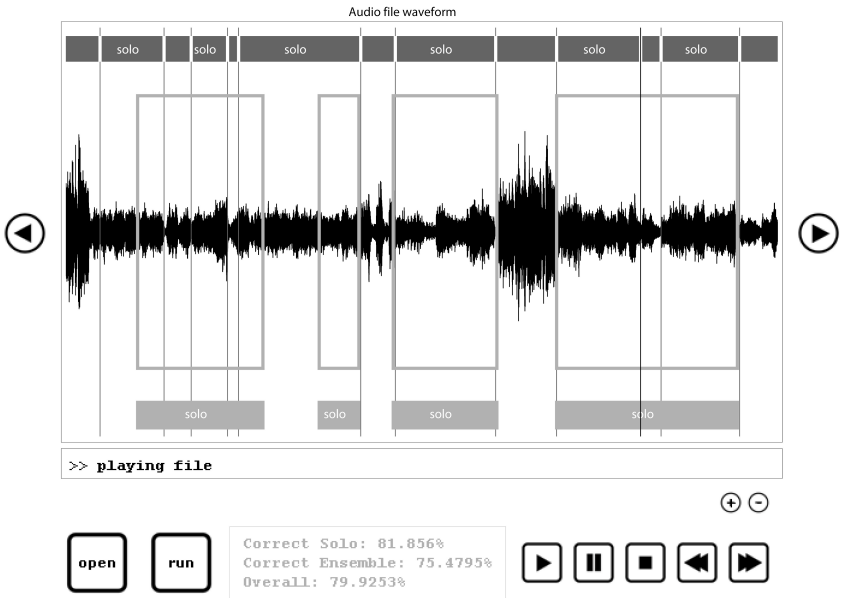

Figure 7: User interface showing the segmentation of a classical piano piece with orchestral accompaniment. The top labels and vertical lines represent the ground-truth whereas the rectangles and bottom labels indicate the extracted solo phrases.

4 Discussion

The results in the preceding section demonstrate that the presented system is able to discriminate between solo and ensemble sections, showing good results for classical music with a final 77% of correct decisions (*CD*). However, experiments on different musical styles such as jazz and popular music, which are not included in this work, showed clear limitations of the current algorithm. The performance of the system degraded when applied to musical pieces from these genres. This implies that the basic assumptions we have made in the definition of the concept *solo* in section 1 should be extended for a more general case. Furthermore, modelling solo sections across different musical styles would require more sophisticated signal descriptors, adapted to their respective playground. We will come back to this issue when presenting further research lines in section 5.

Considering our additional evaluation metrics $CG(p)$, $US(p)$ and $OS(p)$, which capture the quality of the output segmentation with respect to the annotated ground truth, we can observe that oversegmentation occurs more than two times more often than undersegmentation. This implies that a given reference segment is more likely to be split into several output segments, which results in short spurious segments of the opposite class. Comparing the results of the aforementioned metrics for the classifier and the complete system output, we observe that for the $CG(p)$ and $US(p)$ metrics the classifier performs better whilst for the $OS(p)$ the joint segmentation performs better. This is due to the merging of the boundary candidates and classifier output, where a majority vote is used for labelling a sequence of frames between adjacent boundaries. As this procedure produces longer segments than the ones produced by the classifier, undersegmentation errors are more likely to occur whereas oversegmentation is reduced.

Discussing the evaluation procedure itself, one may ask the question of how to objectively evaluate the results of such a system, when the task itself is subject to musical debate. For instance, we could select a specific definition as our objective criterion. The Grove Dictionary of Music and Musicians defines *solo* as: “A piece played by one performer, or a piece for one melody instrument with accompaniment” and “...a section of the composition in which the soloist dominates and the other parts assume a distinctly subordinate role” (Fuller, 2001). While “A piece played by one performer” is quite easy to map to the tasks of our system, mapping “a piece for one melody instrument with accompaniment” and the rest of the definition is not straightforward. If we

listen to classical pieces where the piece is written for an instrument with accompaniment (e.g. violin concerto), we will find sections where the solo instrument stops playing for an ensemble section, and we will find sections where the soloing instrument is dominating and plays a solo with accompaniment. Determining these sections is more or less trivial for both a human listener and our algorithm. But we will also find many sections where the solo instrument is playing and we would not consider these section to be solos. Even if this instrument is more or less predominant in these sections, it is somehow embedded in the accompaniment and plays an ensemble phrase. Thus, in many cases only the musical meaning gives us evidence about the presence of a solo. As we can see our two classes *solo* and *ensemble* are not entirely separated and there exists a "musical greyzone" which is hard to handle, even for human listeners.

5 Conclusions & further work

In this paper we presented a novel system for segmenting and labelling a given music file into solo and ensemble sections. We used local changes of spectral and pitch related features to obtain boundary candidates and assigned labels to the generated segments by using the output of a frame-based classification system. For evaluating the output of the algorithm we used standard precision and recall measurements for the segment boundaries and quantitative, frame-based measurements for the accuracy of the overall system. Furthermore, we introduced novel metrics from image segmentation for a more qualitative evaluation of the output segmentation. We showed that our selected features are instrument-independent and applicable to this segmentation problem. We demonstrated good performance for an evaluation set consisting of full-length classical pieces and achieved accuracies greater by more than 10 percent units compared to existing studies which tackled similar problems.

Currently our system only operates with spectral and pitch related information. Future plans include integrating other cues in order to enhance the general performance and make it more applicable to different kinds of music. Therefore we want to investigate structural cues, taking advantage of the information provided by the repetitions of certain structural segments inside a target song. Particularly western popular music often follows strong structural patterns, which we want to exploit in further research. Moreover, we plan to derive and test additional audio descriptors, which could be able to discriminate between solo and ensemble phrases in all kinds of music. We also want to consider rules of modern music production and use the information provided by the spatial domain for the analysis of western popular music.

Finally, in order to obtain more ground truth data and make further conclusions we want to conduct some listening experiments with the system's output. As we are dealing with technologically motivated definitions of musical concepts, we think that subjective evaluation is a useful tool for making a more general assessment of these musical, thus subjective, issues.

Acknowledgements

The authors want to thank Ricard Marxer for his fruitful comments and suggestions for the boundary estimation. Many thanks also go to Graham Coleman and Justin Salamon for their help in improving the overall quality and style of this publication. This research has been partially funded by the EU-IP project PHAROS[3] IST-2006-045035.

References

Beauchemin, M., & Thomson, K. (1997). The evaluation of segmentation results and the overlapping area matrix. *International Journal of Remote Sensing*, *18*(18), 3895–3899.

Brossier, P. (2006). *Automatic annotation of musical audio for interactive applications*. Unpublished doctoral dissertation, Queen Mary University.

Castleman, K. (1996). *Digital image processing*. Englewood Cliffs, NY, USA: Prentice-Hall.

Chang, C., & Lin, C. (2001). *LibSVM: a library for support vector machines* (Tech. Rep.). Department of Computer Science and Information Engineering, National Taiwan University.

Cheng, S., & Wang, H. (2004). Metric-seqdac: a hybrid approach for audio segmentation. In *Proc. of the 8th international conference on spoken language processing* (pp. 1617–1620).

Cheveigne, A. de, & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, *111*(4), 1917–1930.

[3] http://www.pharos-audiovisual-search.eu

Eggink, J., & Brown, G. J. (2004). Extracting melody lines from complex audio. In *Proc. of the 5th international symposium on music information retrieval* (pp. 84–91).

Foote, J. (2000). Automatic audio segmentation using a measure of audio novelty. In *Proc. of the international conference on multimedia and expo* (Vol. 1, pp. 452–455).

Fuller, D. (2001). Solo. In S. Sadie & J. Tyrrell (Eds.), *The new grove dictionary of music and musicians* (Second ed.). London, UK: Oxford University Press.

Goto, M. (2006). A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech and Language Processing*, *14*(5), 1783–1794.

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. of the international conference on machine learning* (pp. 359–366).

Hollander, M., & Wolfe, D. A. (1999). *Nonparametric statistical methods* (2nd ed.). New York, NY, USA: John Wiley & Sons, Inc.

Izmirili, O. (2000). Using a spectral flatness based feature for audio segmentation and retrieval. In *Proc. of the 1st international symposium on music information retrieval*.

Lu, L., Jiang, H., & Zhang, H. (2001). A robust audio classification and segmentation method. In *Proc. of the 9th ACM international conference on multimedia* (pp. 203–211).

Lu, L., Wang, M., & Zhang, H. (2004). Repeating pattern discovery and structural analysis from acoustic music data. In *Proc. of the 6th ACM SIGMM international workshop on multimedia information retrieval* (pp. 275–282).

Maher, R., & Beauchamp, J. (1994). Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, *95*(4), 2254–2263.

Ong, B., & Herrera, P. (2005). Semantic segmentation of music audio contents. In *Proc. of the international computer music conference* (pp. 61–64).

Orio, N. (2006). Music retrieval: a tutorial and review. In *Foundations and trends in information retrieval* (Vol. 1, pp. 1–96). Hanover, MA, USA: Now Publishers Inc.

Ortiz, A., & Oliver, G. (2006). On the use of overlapping area matrix for image segmentation evaluation: a survey and new performance measure. *Pattern Recognition Letters*, *27*(16), 1916–1926.

Peterschmitt, G., Herrera, P., & Gómez, E. (2001). Pitch-based solo location. In *Proc. of the mosart workshop* (pp. 239–243).

Plomp, R., & Levelt, W. (1965). Tonal consonance and critical bandwidth. *Journal of the Acoustical Society of America*, *38*(4), 548–560.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing. second edition*. Cambridge University Press.

Salembier, P., & Sikora, T. (2002). *Introduction to MPEG-7: Multimedia content description interface* (B. S. Manjunath, Ed.). New York, NY, USA: John Wiley & Sons, Inc.

Smit, C., & Ellis, D. (2007). Solo voice detection via optimal cancellation. In *Proc. of the IEEE workshop on applications of signal processing to audio and acoustics* (pp. 207–210).

Terhardt, E. (1979). Calculating virtual pitch. *Hearing Research*, *1*(2), 155–182.

Turnbull, D., Lanckriet, G., Pampalk, E., & Goto, M. (2007). A supervised approach for detecting boundaries in music using difference features and boosting. In *Proc. of the 8th international symposium on music information retrieval* (pp. 51–54).

Witten, I., Franck, E., Trigg, L., Hall, M., & Holmes, G. (1999). Weka: Practical machine learning tools and techniques with java implementations. In *Proc. of the international workshop on emerging engineering and connectionist-based information systems* (pp. 192–196).

Yang, L., Albregtsen, F., Lonnestad, T., & Grottum, P. (1995). A supervised approach to the evaluation of image segmentation methods. In *Proc. of the 6th international conference on computer analysis of images and patterns* (pp. 759–765).

Zhang, T., & Kuo, C. (2001). Audio content analysis for online audiovisual data segmentation and classification. *IEEE Transactions on Speech and Audio Processing*, *9*(4), 441–457.