Block-Level Audio Features for Music Genre Classification

Klaus Seyerlehner
Dept. of Computational Perception
Johannes Kepler University
Linz, Austria
klaus.seyerlehner@jku.at

Markus Schedl
Dept. of Computational Perception
Johannes Kepler University
Linz, Austria
markus.schedl@jku.at

ABSTRACT

While frame-level audio features, e.g. MFCCs, in combination with the bag-of-frames approach have widely and successfully been used, we use a block processing framework in our submission. In general block-level features have the advantage that they can capture more temporal information than BOF approaches can. We introduce two novel spectral patterns, closely related to the spectrum histogram and propose a modified version of the well-known fluctuation patterns. Based on these patterns we train a support vector machine to classify songs into different categories.

1. AUDIO PREPROCESSING

We use the Java based audio signal analysis toolbox CoMIRVA (Collection of Music Information Retrieval and Visualization Applications) [1]. This library takes care of decode and resample any input audio file to 22 kHz raw PCM. A maximum of four minutes starting from the beginning of an audio file are decoded and the central two minutes of the decoded audio signal are analyzed per audio file. To analyze the audio signal it is transformed to the frequency domain by applying a Short Time Fourier Transform (STFT) using a window size of 2048 samples, a hop size of 512 samples and a Hanning window. Finally, we compute the magnitude spectrum $|X(f)|$ thereof.

1.1 Cent-Scale

We especially account for the musical nature of the audio signals by mapping the magnitude spectrum with linear frequency resolution onto a logarithmical musical scale, the Cent-scale [7].

$$f_{cent} = 1200 \log_2(f_{Hz}/(440 * (\sqrt[1200]{2})^{-5700}))$$

We do so by simply summing all frequency bins of the magnitude spectrum with linear frequency resolution within a constant bandwidth of 100 cent starting from 2050 cent (equal to about 53.43 Hz). The resulting spectral feature vectors still have 97 dimensions. This results in a linear frequency resolution up to about 430 Hz and starts compressing the higher frequency content thereafter in a logarithmic way (see figure 1).

$$X(k)_{dB} = 20 \log_{10}(X(k))$$

We transform the compressed magnitude spectrum $X(k)$ according to the above equation to obtain a logarithmic scale. Altogether, the mapping onto the Cent-scale is a fast approximation of a constant-Q transform, but with constant window length for all frequency bins.

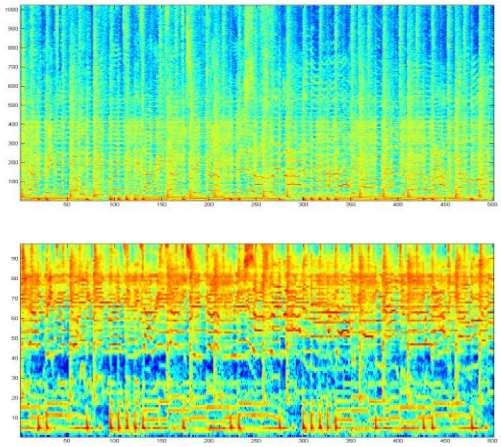

Figure 1 Spectrogram with linear frequency resolution (upper illustration) and the cent-scaled equivalent (lower illustration).

1.2 Audio Normalization

Audio files are recorded at different volume levels. From a technical point of view this means that the whole audio signal $s(t)$ is amplified by a constant factor a

$$\hat{s}(t) = a * s(t)$$

The magnitude spectrum $|\hat{X}(f)|$ of the amplified signal is also scaled by the constant factor a as the Fourier transform is a linear transformation.

$$|\hat{X}(f)| = a * |X(f)|$$

As we process all audio blocks based on a logarithmic amplitude scale (in dB), the amplified magnitude spectrum (in dB) is offset by a constant.

$$\begin{aligned} 20 \log_{10}(|\hat{X}(f)|) &= 20 \log_{10}(\mathrm{a} * |X(f)|) \\ &= 20 \log_{10}(\mathrm{a}) + 20 \log_{10}(|X(f)|) \end{aligned}$$

For some features can be advantageous to be loudness invariant. Thus, we perform an audio normalization. In some audio applications this is achieved by a simple frame by frame mean removal. Removing the mean of each frame of course makes the spectral representation invariant to the constant offset. However, the local loud-

ness information is lost, as all frames will have zero mean. The only information left is the spectral envelope of the audio frame. To keep some local loudness information but still make the whole audio signal loudness invariant the constant offset of a frame is estimated not just based on a single local frame, but using a fixed size neighborhood (in our experiments we use ±100 frames) around each frame. From each frame we remove the mean of its neighborhood.

2. BLOCK PROCESSING

2.1 Block Processing Framework

For block-based audio features the whole spectrum is processed in terms of blocks. Each block consists of a fixed number of spectral frames defined by the *block width*. Two successive blocks are related by advancing in time for a given number of frames specified by the *hop size*. Depending on the *hop size* blocks may overlap or there can even be unprocessed frames between blocks. Figure 2 illustrates how to process an audio file block by block.

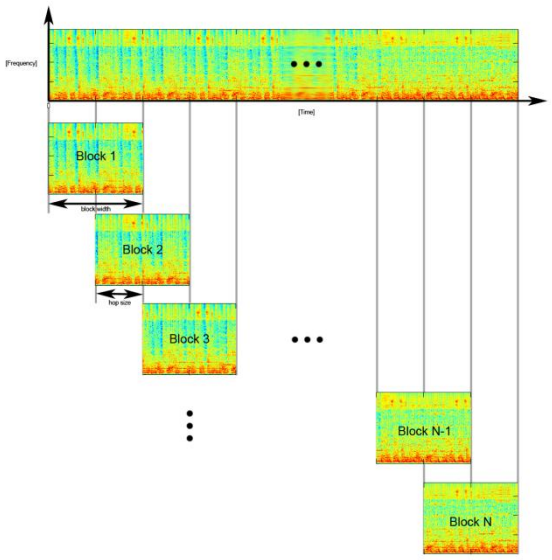

Figure 2 Blockwise processing of the cent spectrum.

2.2 Generalization

To come up with a global feature vector per song, the feature values of all blocks have to be combined into a single descriptor value. To generalize from these local block level feature values we try to pick a representative value for the whole song by picking the value at a specified quantile (e.g. the median) for each feature dimension.

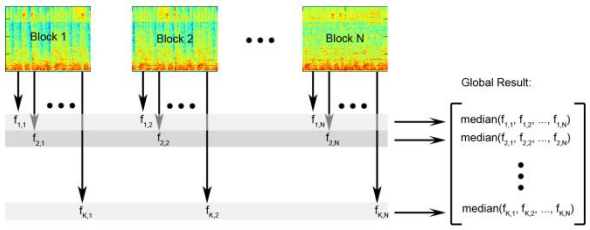

Figure 3 *:* Generalization from block level feature vectors to song feature vectors.

Within this general block processing framework any block level feature is defined with respect to the current block being analyzed. A block can be interpreted as a matrix that has *W* columns defined by the block-width and *H* rows defined by the number of frequency bins.

$$block = \begin{bmatrix} b_{H,1} & \dots & b_{H,W} \\ \vdots & \ddots & \vdots \\ b_{1,1} & \dots & b_{1,W} \end{bmatrix}$$

For the rest of this report, the description of block based audio features, we only outline the block level feature extraction process by explaining how to compute the feature values on block-level.

3. BLOCK-LEVEL AUDIO FEATURES

In our current implementation we use three complex block level features, which we combine by concatenating the resulting feature vectors.

3.1 Modified Fluctuation Patterns

To represent the rhythmic structure of a song we extract a modified variant of the fluctuation patterns proposed by Pampalk et al. [6]. In contrast to the original implementation our variant is not based on the sone representation, but on the cent spectrum. Thus, the extraction of the fluctuation patterns corresponds to the description in [6] starting from step (7) and is based on the cent spectrum instead of the sone representation. Furthermore, no gradient filter is used in our implementation. To generalize from individual blocks the median is used as generalization function.

3.2 Spectral Pattern

To characterize the frequency or timbral content of each song we take short blocks of the cent spectrum containing five frames. Then we simply sort each frequency band of the block.

$$block = \begin{bmatrix} sort(b_{H,1} & \dots & b_{H,W}) \\ \vdots & \ddots & \vdots \\ \text{sort}(b_{1,1} & \dots & b_{1,W}) \end{bmatrix}$$

During the generalization step we use a quantile value of 0.9.

3.3 Delta Spectral Pattern

The delta spectral patterns is basically the same as the spectral pattern, but is not based on the cent-spectrum directly, but on the delta cent-spectrum. The delta cent-spectrum is obtained by subtracting from the original cent spectrum a version that is delayed by some frames (e.g.: five frames). To obtain the delta cent spectrum the result is rectified. Once more the bands of each block are sorted and in the generalization step we take the 0.9 quantile. Figure 4 shows the spectral pattern and the delta spectral pattern of a classical and a hip-hop song.

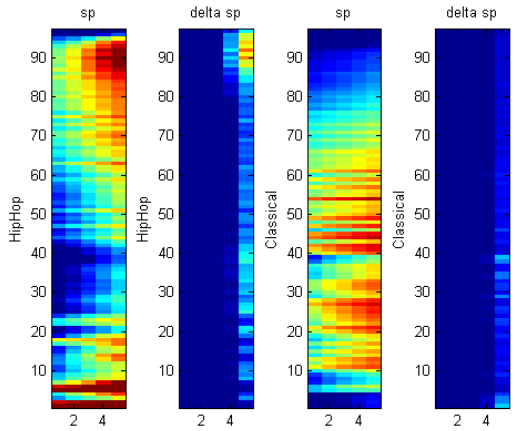

Figure 4 : Generalization from block level feature vectors to song feature vectors.

4. CLASSIFICATION & RESULTS

Using the block-level audio features described in section 3, we train a support vector machine classifier. In our implementation we use the WEKA [2] machine learning toolbox to train and classify songs. The standard settings of the support vector machine classifier of WEKA are used. To evaluate our approach it has been tested using ten times 10-fold cross validation on two well-known public datasets. We report the mean and the standard deviation of the classification accuracies obtained by the individual runs of the 10-fold cross validation. On the ISMIR 2004 genre [4] classification dataset containing 1458 full length audio recordings we obtain a classification accuracy of **82.72 (0.68)%.** On the GTZAN dataset [5] consisting of 1000 30 seconds excerpts we obtain a classification accuracy of **77.96 (0.65)%.** Comparing these results to those obtained by other systems (see table 1) we conclude that the implemented block-level music classification system performs comparably to state-of-the-art systems.

Reference	Dataset	Accuracy
Bergstra et. al.	GTZAN	82.50%
Li. et al.	GTZAN	78.50%
Lidy et al.	GTZAN	76.80%
Benetos et al.	GTZAN	75.00%
Holzapfel et al.	GTZAN	74.00%
Tazanetakis et al.	GTZAN	61.00%
Holzapfel et al.	ISMIR2004	83.50%
Pampalk et al.	ISMIR2004	82.30%
Lidy et al.	ISMIR2004	79.70%

Table 1. Notable classification accuracies achieved by music genre classification approaches (see [4]).

5. REFERENCES

[1] M. Schedl, P. Knees, K. Seyerlehner, T. Pohle. The CoMIRVA Toolkit for Visualizing Music-Related Data. *Proc. of the 9th Eurographics/IEEE VGTC Sym. on Visualization (EuroVis 2007), Norrköping, Sweden, May 2007.*

[2] I. H. Witten, E. Frank. *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, 2005.

[3] K. Seyerlehner, G. Widmer, P. Knees: Frame-level Audio Similarity – A Codebook Approach. In *Proc. of the 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008

[4] I. Panagakis, E. Benetos, C. Kotropoulos: Music Genre Classification: A Multilinear Approach. In *Proc. of the 9th Int. Conf. on Music Information Retrieval (ISMIR'08)*, Philadelphia, USA, 2008.

[5] G. Tzanetakis, P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Audio and Speech Processing*, vol. 10, no. 5, July, 2002.

[6] E. Pampalk, A. Rauber, D. Merkl. Content-based Organization and Visualization of Music Archives, In *Proc. of the ACM Multimedia*, Juan-les-Pins, France, 2002.

[7] M. Goto. Smartmusickiosk: Music listening station with chorus-search function. In *Proc. of the 16th Annual ACM Symp. on User Interface Software and Technology (UIST'03)*, Vancouver, Canada 2003.