FUSING BLOCK-LEVEL FEATURES FOR MUSIC SIMILARITY ESTIMATION

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

Gerhard Widmer
Dept. of Computational Perception,
Johannes Kepler University
Linz, Austria
gerhard.widmer@jku.at

Tim Pohle
Dept. of Computational Perception,
Johannes Kepler University
Linz, Austria
tim.pohle@jku.at

ABSTRACT

In this paper we present a novel approach to computing music similarity based on block-level features. We first introduce three novel block-level features — the *Variance Delta Spectral Pattern* (VDSP), the *Correlation Pattern* (CP) and the *Spectral Contrast Pattern* (SCP). Then we describe how to combine the extracted features into a single similarity function. A comprehensive evaluation based on genre classification experiments shows that the combined block-level similarity measure (BLS) is comparable, in terms of quality, to the best current method from the literature. But BLS has the important advantage of being based on a vector space representation, which directly facilitates a number of useful operations, such as PCA analysis, k-means clustering, visualization etc. We also show that there is still potential for further improve of music similarity measures by combining BLS with another state-of-the-art algorithm; the combined algorithm then outperforms all other algorithms in our evaluation. Additionally, we discuss the problem of album and artist effects in the context of similarity-based recommendation and show that one can detect the presence of such effects in a given dataset by analyzing the nearest neighbor classification results.

1. INTRODUCTION

Although music similarity is naturally an ill-defined concept, as individuals might judge the perceived similarity of two songs differently, it is also a well accepted fact that estimating perceived similarities between songs can be very useful in many applications like music recommendation or automatic playlist generation. Furthermore, new innovative interfaces to explore and organize music collection can be built based on music similarity information. There are two fundamentally different approaches to estimating music similarity. One is to use meta information about music, collected from users, experts, or the Web. The other approach, the so-called content-based approach, is to analyze the music signals themselves. There has been a lot of research on content-based music similarity estimation and the related task of automatic genre classification. A good overview of state-of-the-art systems can be gained by taking a look at the descriptions of the systems submitted to the ***M**usic **I**nformation **R**etrieval **E**valuation e**X**change* (MIREX) [1] 2009 Audio Music Similarity and Retrieval task. Most of the participating algorithms contain components that are based on the so-called Bag of Features (BOF) representation, which typically characterizes the timbral content of an audio file by modeling the distribution of local audio features, e.g., MFCCs. A perfect example is the algorithm proposed by Pohle et al. [1], which ranked first at the MIREX 2009 competition. The corresponding similarity model consists of two distribution models, one describing the rhythmical and the other describing the timbral nature of a song.

In this paper we present a novel approach to computing content-based music similarity, where the individual components are based on a *vector space representation*. The merit of a vector space representation is that a whole toolbox of mathematical methods like, e.g., Principal Components Analysis, k-Means Clustering or Random Projection becomes applicable so that one can easily analyze, compress, visualize, cluster, or learn from the extracted song models. The individual components of the proposed similarity measure are so-called block-level feature. We will start with a short introduction to the block processing framework in section 2. We then introduce three novel block-level audio features in section 3 and describe in section 4 how to combine the described patterns into a similarity measure. In section 5 we evaluate the proposed similarity measure via nearest neighbor music genre classification. We start with a discussion on the evaluation of content-based similarity measures in the context of music recommendation and discuss the impact of artist and album effects. We then present a comparative evaluation of the proposed algorithm and three other music similarity measure, including the state-of-the-art measure by Pohle et al. [1]. We can show that the proposed algorithm is comparable in terms of quality to the state of the art, and that a combination of the block-level similarity measure (BLS) with the measure in [1] outperforms all other algorithms. Finally, we show that one can verify the expected artist/album effect on three genre classification datasets that are used in the evaluation.

[1] http://www.music-ir.org/mirexwiki

2. THE BLOCK PROCESSING FRAMEWORK

The idea of processing audio block by block is inspired by the feature extraction process described in [2, 3, 4]. However, instead of just computing rhythm patterns or fluctuation patterns from the audio signal one can generalize this approach and define a generic framework. Based on this framework one can then compute other features (e.g., the ones presented in section 3) to describe the content of an audio signal. One advantage of block-level features over frame-level features like, e.g., MFCCs is that each block contains a bunch of frames, which allows the extracted features to better capture temporal information. The basic block processing framework can be subdivided into two stages: first, the block processing stage and second, the generalization step.

2.1. Block Processing

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 size. Two successive blocks are related by advancing in time by 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 in between the blocks. Although the hop size could also vary within a single file to reduce aliasing effects, here we only consider constant hop sizes. Figure 1 illustrates how to process an audio file block by block.

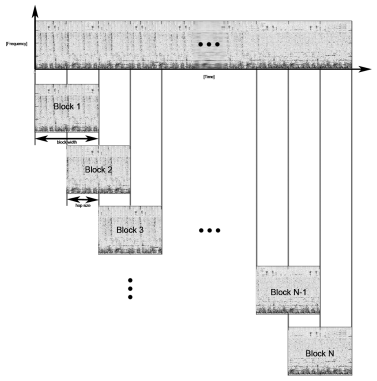

Figure 1: *Block by block processing of the cent spectrum.*

2.2. Generalization

To come up with a global feature vector per song, the feature values of all blocks must be combined into a single representation for the whole song. To combine local block-level feature values into a model of a song, a summarization function is applied to each dimension of the feature vectors. Typical summarization functions are, for example, the mean, median, certain percentiles, or the variance over a feature dimension. Interestingly, also the classic Bag of Frames approach (BOF) (see 5.3.1) can be interpreted as a special case within this framework. The block size would in this case correspond to a single frame only, and a Gaussian Mixture Model would be used as summarization function. However, we do not consider distribution models as summarization functions here, as our goal is to define a song model whose components can be interpreted as vectors in a vector space. The generalization process is illustrated in Fig. 2 for the median as summarization function.

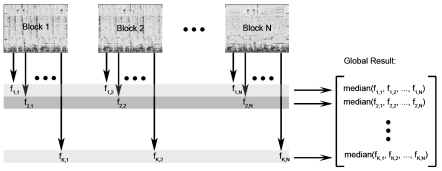

Figure 2: *Generalization from block level feature vectors to song feature vectors using the median as summarization function.*

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 frequency resolution (the number of frequency bins).

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

For the rest of this paper, the description of block-based audio features, we only outline the block-level feature extraction process by describing how to compute the feature values on a single block and by describing the generalization function that is used.

3. BLOCK-LEVEL FEATURES

3.1. Audio Preprocessing

All block-level features presented in this paper are based on the same spectral representation: the cent-scaled magnitude spectrum. To obtain this, the input signal is downsampled to 22 kHz and 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. Then we compute the magnitude spectrum $|X(f)|$ thereof and account for the musical nature of the audio signals by mapping the magnitude spectrum with linear frequency resolution onto the logarithmic Cent scale [5] given by Equation (2). In our implementation this results in a linear frequency resolution for the lower bands and then starts to compress the higher frequency bands in a logarithmic way.

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

The compressed magnitude spectrum $X(k)$ is then transformed according to Eq.3 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 size for all frequency bins.

$$X(k)_{dB} = 20 \log_{10}(X(k)) \tag{3}$$

Finally, we perform a normalization step such that the obtained spectrum is intensity-invariant by removing the mean computed over a sliding window from each audio frame as described in [6]. All features presented in the next section are based on the normalized cent spectrum.

3.2. Feature Extraction

In this section we describe the block-level features that will be used to describe audio content. Three of these features — *Spectral Pattern* (SP), *Delta Spectral Pattern* (DSP) and the *Fluctuation Pattern* (FP) — have already been used in our MIREX 2009 submission [6]. In contrast to [6] we will use a logarithmically scaled variant of the FP, the *Logarithmic Fluctuation Pattern* (LFP), in this paper. The other three features — *Variance Delta Spectral Pattern* (VDSP), *Correlation Pattern* (CP) and *Spectral Contrast Pattern* (SCP) — are novel features and are presented here for the first time. The parameter setting presented here was obtained via optimization, which is described in more detail in Section 4.2.

3.2.1. Spectral Pattern (SP)

To characterize the frequency or timbral content of each song we take short blocks of the cent spectrum containing 10 frames. A hop size of 5 frames is used. Then we simply sort each frequency band of the block.

$$\mathrm{SP} = \begin{bmatrix} \mathrm{sort}(b_{H,1} & \cdots & b_{H,W}) \\ \vdots & \ddots & \vdots \\ \mathrm{sort}(b_{1,1} & \cdots & b_{1,W}) \end{bmatrix} \tag{4}$$

As summarization function the 0.9 percentile is used.

3.2.2. Delta Spectral Pattern (DSP)

The Delta Spectral Pattern is extracted by computing the difference between the original cent spectrum and a 3 frames delayed version of the cent spectrum to emphasize onsets. The resulting delta spectrum is rectified so that only positive values are kept. Then we proceed exactly as for the Spectral Pattern and sort each frequency band of a block. A block size of 25 frames and a hop size of 5 frames are used, and the 0.9 percentile serves as summarization function. It is important to note that the DSP's block size differs from the block size of the SP; both were obtained via optimization. Consequently, the SP and the DSP capture information over different time spans.

3.2.3. Variance Delta Spectral Pattern (VDSP)

The feature extraction process of the Variance Delta Spectral Pattern is the same as the feature extraction process of the Delta Spectral Pattern (DSP). The only difference is that the Variance is used as summarization function over the individual feature dimensions. While the Delta Spectral Pattern (DSP) tries to capture the strength of onsets, the VDSP should indicate if the strength of the onsets varies over time or, to be more precise, over the individual blocks. A hop size of 5 and a block size of 25 frames are used.

3.2.4. Logarithmic Fluctuation Pattern (LFP)

To represent the rhythmic structure of a song we extract the Logarithmic Fluctuation Patterns, a modified version of the Fluctuation Pattern proposed by Pampalk et al. [2]. A block size of 512 and a hop size of 128 are used. We take the FFT for each frequency band of the block to extract the periodicities in each band. We only keep the amplitude modulations up to 600 bpm. The amplitude modulation coefficients are weighted based on the psychoacoustic model of the fluctuation strength according to the original approach in [2]. To represent the extracted rhythm pattern in a more tempo invariant way, we then follow the idea in [1, 7, 8] and represent periodicity in log scale instead of linear scale. Finally, we blur the resulting pattern with a Gaussian filter, but for the frequency dimension only. The summarization function is the 0.6 percentile.

3.2.5. Correlation Pattern (CP)

To extract the Correlation Pattern the frequency resolution is first reduced to 52 bands. This reduction of the frequency resolution was found to be useful by optimization and also reduces the dimensionality of the resulting pattern. Then we compute the pairwise linear correlation coefficient (Pearson Correlation) between each pair of frequency bands. The result of this operation is a symmetric correlation matrix. The Correlation Pattern can capture, for example, harmonic relations of frequency bands in the case that sustained musical tones are present. Also rhythmic relations can be reflected by the correlation pattern. For example, if a bass drum is always hit simultaneously with a high-hat this would result in a strong positive correlation between low and high frequency bands. Conversely, if the high-hat and the bass drum are never played together this would contribute to a negative correlation. The CP shows interesting patterns when they are visualized for different types of songs. For example the presence of a singing voice leads to very specific correlation patterns. A block size of 256 frames and a hop size of 128 frames is used. The summarization function for this feature is the 0.5 percentile.

3.2.6. Spectral Contrast Pattern (SCP)

The Spectral Contrast [9] is a feature that roughly estimates the *"tone-ness"* of a spectral frame. This is realized by computing the difference between spectral peaks and valleys in several sub-bands. As the strong spectral peaks roughly correspond to tonal components and flat spectral excerpts roughly correspond to noise.like or percussive elements, the difference between peaks and valleys characterizes the toneness in each sub-band. In our implementation the Spectral Contrast is computed based on the cent scaled spectrum subdivided into 20 frequency bands. For each frame we compute in each band the difference between the maximum value and the minimum value. This results in 20 Spectral Contrast values per frame. Then the Spectral Contrast values of a block are sorted within each frequency band, as already described for the Spectral Pattern. A block size of 40 frames and a hop size of 20 frames are used. The summarization function is the 0.1 percentile. Figure 3 visualizes the proposed set of features for two different songs, a Hip-Hop and a Jazz song.

3.3. Merits of the vector space model

To further emphasize that a vector space representation has indeed some important merits in contrast to distribution models, we have computed a *Principal Component Analysis* (PCA) for the *Spectral Pattern* (SP) based on the *"1517-Artists"* dataset (see 5.2.5). The first 13 principal components are visualized in Figure 4. The components seem to capture different musical concepts like bass elements, drum elements or different tones. Furthermore, the first 110 components capture 99.01% of the total variance. This indicates that there is a lot of redundant information in the Spectral Patterns and that the SP can be radically compressed without any substantial loss in quality. Although this is just one example of the benefits of a vector space representation and further analysis of

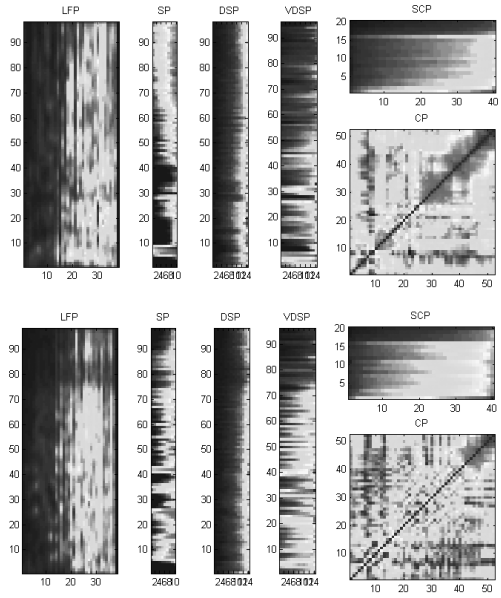

Figure 3: *Visualization of the proposed block-level patterns for a Hip-Hop song (upper) and a Jazz song (lower).*

the features is left for future work, we believe that a vector space representation can be quite beneficial especially because of all the rich mathematical methods available for vector spaces.

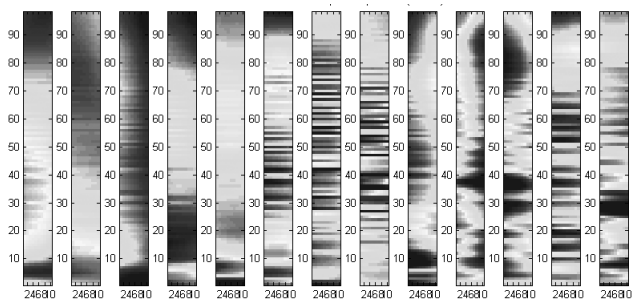

Figure 4: *Visualization of the 13 most important principal components of the Spectral Pattern.*

4. FUSING DISTANCES FOR MUSIC SIMILARITY ESTIMATION

To define a single similarity function based on the six block-level patterns presented above, we first define a similarity function for the individual patterns alone. In our case we get good genre classification results for all individual patterns when they are compared using the *Manhattan distance* (l_1 norm) as a similarity function. In a next step the distances resulting from the individual components have to be combined into a single similarity measure. One main problem is that the similarity estimates based on the individual patterns have different scales. To account for this, we follow

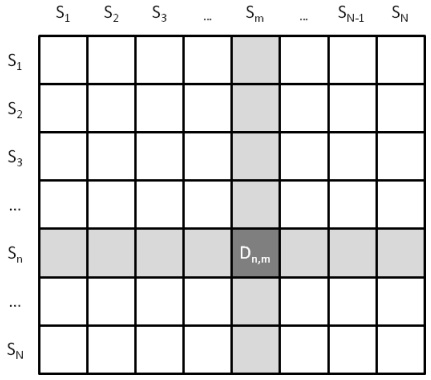

Figure 5: *Distance Space Normalization of a distance matrix.*

the approach in [10] and perform a distance space normalization (DSN). However, we will use a modified variant that allows for a more intuitive interpretation. Each distance of a distance matrix $D_{n,m}$ is normalized by subtracting the mean and dividing by the standard deviation (Gaussian normalization) over all distances in row n and column m (see figure 5). Thus, each distance between two songs n and m has its own normalization parameters, as all distances to song m and all distances to song n are used for normalization. This way the normalization operation can also change the ordering within a column / row, which can even have a positive influences on the nearest neighbor classification accuracy according to [1]. This observation is confirmed by the classification results of the individual components of our approach before and after distance space normalization summarized in Table 1.

After the distances resulting from the individual components have been normalized they are summed to yield an overall similarity measure. The results in Table 1 also shows that this combination clearly outperforms the individual components. Additionally, it turns out that removing any of the proposed components would reduce the classification accuracy of the combined similarity measure. Thus, we can conclude that all the components contribute to the overall similarity. It is also worth mentioning that we have evaluated several other normalization approaches to combine the individual components, but none outperformed the DSN approach described here.

4.1. Extended Distance Space Normalization (EDSN)

It is straightforward to extend this additive combination approach by assigning individual weights to each component. Additionally, based on the observation that the DSN approach all alone can help to improve nearest neighbor classification accuracy, we extend the combination approach and once more normalize the resulting distance matrix after the individually weighted components have been combined. This results in an improvement of the classification accuracy from 36.49% to 37.69% (see Table 1). The overall combination method we propose is visualized in Figure 6. In the next subsection we will discuss how to obtain the component weights and the parameter settings for the presented features via optimization.

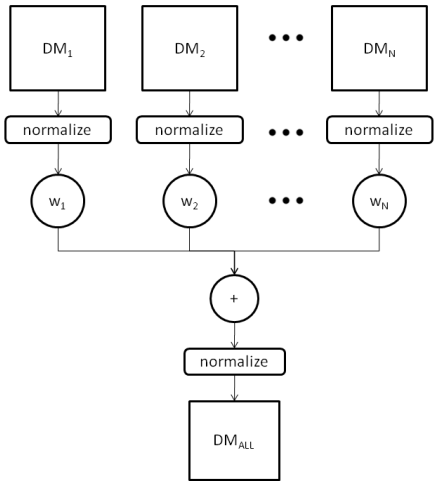

Figure 6: *Schematic structure of the combination approach.*

4.2. Optimization of the Similarity Measure

The features presented in section 3.2 come with a large set of parameters. For example, one can compute these features for different block sizes, hop sizes, different percentiles and other parameters. To identify suitable parameter settings for the individual components and find an optimal weighting of the components we optimize these settings on dataset *"1517-Artists"* (see 5.2.5).

Each block-level feature was optimized separately. Unfortunately, a complete evaluation of all reasonable parameter combinations was impossible, as each parameter setting requires to recompute the feature on the whole collection, which was not tractable computationally. Therefore, the optimization procedure was to vary each parameter separately one after the other and pick the optimum. The *"optimal"* parameter settings found by this still expensive optimization are reported in Section 3.2.

The weights of the individual components are found by a greedy optimization. All the components get an equal weight at the beginning ($w_i = 1$). Then the weight of a randomly chosen component is reduced, if this results in an improvement in classification accuracy. This step is repeated until no more improvement is possible. The final weights that result from this optimization are summarized in Table 1.

Component	Weight	10-NN	10-NN (DSN)
LFP	1.0	26.70%	28.75%
SP	1.0	26.76%	26.82%
SCP	1.0	19.31%	20.47%
CP	1.0	23.36%	26.83%
VDSP	0.8	21.44%	24.18%
DSP	0.9	23.18%	25.28%
Combined		36.49%	37.69%

Table 1: *Component weights of the optimized similarity measure and the respective nearest neighbor classification accuracies on dataset "1517-Artists" of the components before and after distance space normalization (DSN).*

5. EVALUATION

To evaluate the proposed block-level similarity measure (BLS) we follow the standard procedure in MIR research and evaluate the measure in an indirect way via nearest neighbor music genre classification. In our evaluation we especially take into account the intended target application of the proposed similarity measure, which is automatic music recommendation. Thus, we start with a discussion on the evaluation of content-based similarity measures in the context of music recommendation in subsection 5.1. Then, in 5.2, we introduce the six genre classification datasets that are used in the evaluations and briefly describe, in 5.3, the three similarity measures that the BLS approach is compared to. Finally, the results of the evaluation are presented and analyzed in subsection 5.4.

5.1. Evaluation in the context of Music Recommendation

The most widespread application of a content-based similarity measure is that one can easily build a music recommendation engine simply by recommending the songs that sound most similar to a given query song according to the similarity measure. However, not all songs that *"sound similar"* are also good recommendations. It is for example quite obvious that users would be annoyed if just songs by the same artist as the query song or by just one other artist appear in the recommendations. Unfortunately, this is often the case if the recommendations are generated with a content-based similarity measure: songs by one artist are of course often very similar to each other, because of the very specific style and the specific recording conditions. From an acoustic point of view the characteristic style of an artist is related to the instrumentation, the singing voice and the type of music being played, which are all in some way reflected in a good similarity measure. Thus, it is relatively easy to identify songs by one and the same artist using audio similarity algorithms. This effect is known as *artist effect*. In some cases even album-specific production effects are reflected in the spectral representation of songs, which is respectively called *album effect*. Artist and album effects are not a problem *per se*, but they do become one in the context of music recommendation, and also when we use – as is generally done in MIR research – *genre classification* as an indirect method for evaluating the quality of music similarity measures: clearly, songs by the same artist will tend to belong to the same genre, and the ability to recognize the genre by, e.g., specific production effects, is not what we intend to measure.

For an informative evaluation, that implies that trivial similarity recognition should be prevented by using a so-called *artist filter* that ensures that all songs by one artist are either in the training set, or in the test set. But although album and artist effects have been studied intensively in the literature [11], often evaluation results without artist filtering are reported. One reason might be that there are almost no publicly available datasets that contain songs by a sufficiently large number of different artists. The only two datasets that we are aware of that might be useful for evaluations in this context are the *"Homburg"* [12] and the *"1517-Artists"* dataset. Unfortunately, the song excerpts in the former are only 10 seconds long, and the latter has been used to optimize the proposed block-level similarity measure. For this reason we have compiled another additional dataset, called *"Unique"*. For better comparability we will also present nearest neighbor classification results on three well-known and publicly available datasets, where we ex-

pect a significant artist effect – which will also be experimentally verified.

5.2. Datasets

Six different genre classification datasets are used for the evaluation. Three of these (*"GTZAN"*, *"ISMIR 2004 Genre"* and *"Ballroom"*) are well-known datasets and should help to make the evaluation results comparable to previous results in the literature. However, we suspect that there is a significant artist effect in these three datasets. For the other three datasets (*"Homburg"*, *"1517-Artists"* and *"Unique"*), artist information is available and we use an artist filter to prevent any artist or album effects.

5.2.1. GTZAN

The GTZAN dataset contains 1000 song excerpts (30 seconds) evenly distributed over 10 genres. Unfortunately, there is no artist information available and the number of different artists is unknown. However, we expect an artist effect in this dataset, as listening to some of the songs reveals that some artists are represented with several songs.

5.2.2. ISMIR 2004 Genre

The ISMIR 2004 Genre dataset was used as ground truth in the first public evaluation of content-based algorithms. There are two different versions of the dataset. In our evaluation we will use the larger dataset consisting of 1458 full length tracks distributed over 6 genres. The class distribution is unequal. The dominating class "classical", comprising 43.9% of all songs. Artist information is only partially available, but the number of different artists is rather low and consequently a strong artist effect is to be expected.

5.2.3. Ballroom

The ballroom collection consists of 698 song excerpts (30 seconds) of 8 different dance music styles (Cha-cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz and Waltz). The genre distribution is quite uniform. The category with the fewest examples is represented by 8.6%, the largest category by 15.9% of all examples. Artist information is missing, but for a part of the songs album information is available. We expect that there exists an album or artist effect on this dataset, since there are often many examples that belong to one album.

5.2.4. Homburg

The *"Homburg"* dataset [12] contains 1886 songs by 1463 different artists. Consequently, the artist effect should be relatively small, but still we will use an artist filter on this dataset since artist information is available. The rather short song excerpts of 10 seconds length are unequally distributed over 9 genres. The largest class contains 26.72%, the smallest class 2.49% of all songs.

5.2.5. 1517-Artists

This genre classification dataset consists of freely available songs from download.com[2] and has been previously used in [13] and [14]. To ensure reasonable track quality approximately the 190 most popular songs (according to the number of total listenings) were selected for each genre. Altogether there are 3180 tracks by 1517 different artists distributed over 19 genres. It is worth mentioning that this collection has an almost uniform genre distribution, contains tracks from a large number of different artists, and can be freely downloaded from the author's personal homepage[3]. This dataset was used to optimize the parameters of the BLS approach; thus, the classification results reported below are likely to be too optimistic.

5.2.6. Unique

Dataset *"Unique"* contains 30 seconds song excerpts from 3115 popular and well-known songs distributed over 14 genres. The dataset is compiled in such a way that no two songs by one and the same artist are in the dataset. Consequently, there is no need to apply an artist filter. The class distribution is skewed. The smallest class accounts for 0.83%, the largest class for 24.59% of all songs.

5.3. Music Similarity Algorithms

In our evaluation we compare six different approaches: the proposed block-level similarity measure (BLS), the by now 'classic' Single Gaussian Approach (SG), two state-of-the-art methods (RTBOF, MARSYAS), and a combination (CMB) of RTBOF and BLS. Additionally, a random similarity measure (RND) is used to indicate the baseline accuracy for the various datasets. The SG, RTBOF, MARSYAS and CMB measures are briefly described in the following.

5.3.1. Single Gaussian (SG)

The *Single Gaussian* (SG) model is based on the so-called Bag of Frames (BOF) approach [15]. Each song is modeled as a distribution – in the form of a single multi-variate Gaussian – of Mel Frequency Cepstrum Coefficients (MFCCs) over the entire song. The similarity between two such models is computed via the Kullback-Leibler (KL) divergence. This approach is a fast and popular variant proposed by Levy et al. [16] of the classic timbre-based audio similarity measure.

5.3.2. Rhythm Timbre Bag of Features (RTBOF)

The *Rhythm-Timbre Bag of Features* approach (RTBOF-NN) is a recent music similarity measure proposed by Pohle et al. in [1]. This measure ranked first in the MIREX 2009 music similarity and retrieval task and has proven to be statistically significantly better than most of the participating algorithms. Thus, it reflects the current state of the art in music similarity estimation and nearest neighbor classification. Basically, it has two components – a rhythm and a timbre component. Each component consists of a distribution model over local spectral features. The features, described in [1], are complex and incorporate local temporal information over several frames. Because of its components we will call this approach Rhythm-Timbre Bag Of Features (RTBOF) in the following.

[2] http://music.download.com/

[3] www.seyerlehner.info

5.3.3. Marsyas (MARSYAS)

The MARSYAS (Music Analysis, Retrieval and Synthesis for Audio Signals) framework[4] is an open source software that can be used to efficiently calculate various audio features. For a detailed description of the extracted features we refer to [17]. This algorithm has participated in the MIREX Audio Similarity and Retrieval task in 2007 and 2009 (there was no such task in 2008), taking the 2nd and 7th rank, respectively. The features as well as the similarity computation were the same in 2007 and 2009. We use the framework to extract the features and compute the distance matrix exactly as for the MIREX contest.

5.3.4. Combination (CMB)

To find out if there is any potential for further improvement, we also evaluate a combination of the RTBOF and the BLS approach. The similarity measures are combined using the extended distance space normalization presented in Section 4.1, with equal weights for both algorithms.

5.4. Evaluation Results

Genre classification experiments were performed in the form of stratified 10-fold cross-validation runs, using a *k Neareast Neighbor (k-NN)* classifier based on the respective similarity measure, with k varying from 1 to 20. The results are summarized in Figure 7. First of all, all similarity algorithms clearly outperform the random baseline. Note that the random baseline stays constant for datasets *"1517-Artists"* and *"GTZAN"*, while it increases on the other datasets. This increase is related to the imbalanced class distributions of the respective datasets. With increasing k, the random baseline grows towards the accuracy obtainable by always predicting the most frequent genre.

Furthermore, comparing the accuracies obtained on the datasets in the upper row of Figure 7 to the accuracies on the datasets in the lower row, one can see that the classification accuracies are basically increasing in the upper plots and decreasing in the lower plots. In the upper datasets there is no artist or album effect (due to the nature of the music collections), whereas for the datasets in the lower plots we expect an artist effect. This phenomenon can be explained by the fact that for datasets with an artist effect the first nearest neighbor of a given query song will likely be a song by the same artist. As typically songs by the same artist belong to the same genre the first nearest neighbor is an almost perfect genre predictor. For the datasets without artist effect, we can clearly observe the expected learning curve, where the k-NN classifier profits from considering an increasing neighborhood. Thus, from an analysis of the nearest neighbor classification results one can even detect the presence of a strong artist or album effect.

Considering the individual results of the algorithms it turns out that both the RTBOF and the BLS outperform the SG and MARSYAS measures. For the artist-filtered collections (*"1517-Artists"*, *"Homburg"*, *"Unique"*) the BLS measure exceeds the classification accuracy of the RTBOF approach (by 2.31, 1.04 and 0.72 percentage points). On the not-artist-filtered collections (that are *"GTZAN"*, *"Ballroom"*,*"ISMIR 2004 Genre"*), RTBOF performs better than BLS (by 2.56, 7.6 and 0.59 percentage points). One might speculate that the RTBOF approach is more successful in artist identification, whereas the BLS is more suitable in the recommendation context. The only significant difference in classification accuracy on the *"Ballroom"* dataset seems to be also related to the improved variant of the "Fluctuation Pattern" (the "Onset Pattern" (OP)) that are used in RTBOF algorithms. Thus, it might be worth replacing the LFP in our BLS algorithm with the OP, as the OP would also fit into the block-level feature extraction framework and has a vector space representation. Altogether, the differences in classification accuracy between RTBOF and BLS are marginal, and we conclude that both approaches perform comparably in terms of quality.

However, the evaluated *combination approach (CMB)* is clearly an improvement over both the RTBOF and BLS methods. It outperforms all other algorithms on datasets *"1517-Artists"*, *"Homburg"*, *"Unique"* and *"GTZAN"*, and achieves an accuracy comparable to the best single algorithm on datasets *"Ballroom"* and *"ISMIR 2004 Genre"*. This indicates that there is further potential for improvements of the block-level similarity approach, e.g., by replacing the LFP with the OP, but also for content-based similarity algorithms in general. Furthermore, this also indicates that both RTBOF and BLS capture some complementary similarity information. Thus, a more detailed analysis of the individual components of both methods seems called for, to identify an optimal combination.

6. CONCLUSIONS & FUTURE WORK

In this paper we have presented three novel block-level audio features, namely the *Variance Delta Spectral Pattern* (VDSP), the *Correlation Pattern* (CP) and the *Spectral Contrast Pattern* (SCP). We have defined a music similarity measure based on these and three other block-level features and have shown that it is comparable to the state of the art in music similarity. The advantage of the proposed measure is that its components are based on a simple vector representation. This allows to, e.g., easily analyze, compress or cluster the components via standard mathematical procedures. Another advantage is that one can visualize a song's model, and the proposed feature set can be used in a straight-forward way with more powerful classification methods such as, e.g., Support Vector Machine, for other classification tasks too. Thus, one future direction will be to use the proposed feature set in automatic tag prediction. Last, but not least, the evaluated combination approach indicates that combining the work in [1], especially the *"Onset Pattern"*, with some of the proposed block-level patterns could further improve the similarity measure. This has to be investigated in more detail to see which combination of components is most suitable. With respect to the evaluation of music similarity measures we have shown that there exists an artist effect on three well-known datasets and we have presented another novel dataset that is more suitable for evaluations in this context.

7. ACKNOWLEDGMENTS

This research was supported by the Austrian Research Fund (FWF) under grant L511-N15.

8. REFERENCES

[1] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On rhythm and general music similarity," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR-09)*, 2009.

[4] http://marsyas.info

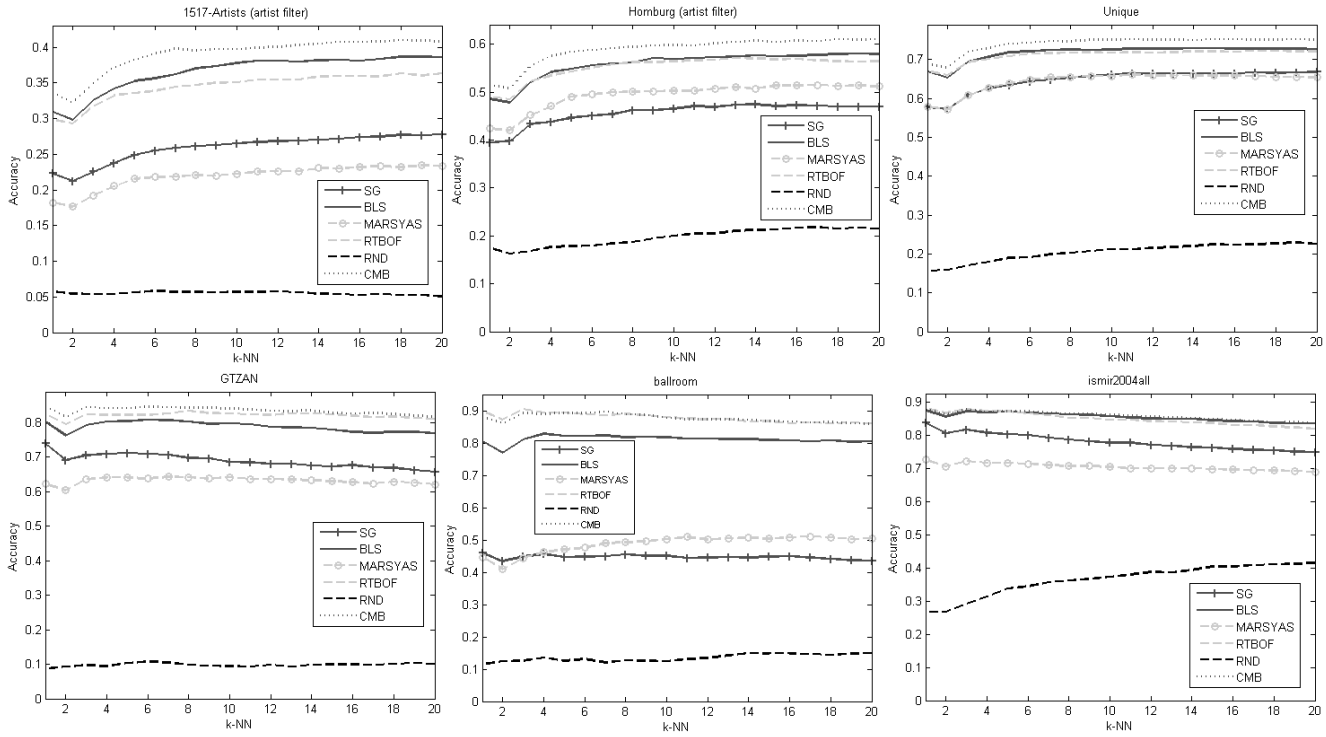

Figure 7: *Evaluation results of nearest neighbor genre classification on six different genre classification datasets.*

[2] E. Pampalk, A. Rauber, and D. Merkl., "Content-based organization and visualization of music archives," in *Proc. of the 10th ACM Int. Conf. on Multimedia*, 2002.

[3] A. Rauber, E. Pampalk, and D. Merkl, *The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models.*, Journal of New Music Research, 2003.

[4] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification.," in *Proc. of the 6th Int. Conf. on Music Information Retrieval (ISMIR-05)*, 2005.

[5] M. Goto, "Smartmusickiosk: Music listening station with chorus-search function.," in *Proc. of the 16th ACM Symp. on User Interface Software and Technology (UIST-03)*, 2003.

[6] K. Seyerlehner and M. Schedl, "Block-level audio feature for music genre classification," in *online Proc. of the 5th Annual Music Information Retrieval Evaluation eXchange (MIREX-09)*, 2009.

[7] J. H. Jensen, M. G. Christensen, and S.H Jensen, "A tempo-insensitve representation of rhythmic patterns," in *Proc. of the 17th European Signal Processing Conf. (EUSIPCO-09)*, 2009.

[8] A. Holzapfel and Y. Stylianou, "A scale transform based method for rhythmic similarity of music," in *Proc. of the 2009 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-09)*, 2009.

[9] D. Jiang, L. Lu, H. Zhang, J. Tao, and L. Cai, "Music type classification by spectral contrast feature," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME-02)*, 2002.

[10] T. Pohle and D. Schnitzer, "Striving for an improved audio similarity measure," in *online Proc. of the 4th Annual Music Information Retrieval eXchange (MIREX-07)*, 2007.

[11] A. Flexer and D. Schnitzer, "Album and artist effects for audio similarity at the scale of the web," in *Proc. of the 6th Sound and Music Computing Conference (SMC-09)*, 2009.

[12] H. Homburg, I. Mierswa, B. Mller K. Morik, and M. Wurst, "A benchmark dataset for audio classification and clustering," in *Proc. of the 6th Int. Sym. on Music Information Retrieval (ISMIR-05)*, 2005.

[13] K. Seyerlehner, G. Widmer, and P. Knees, "Frame level audio similarity - a codebook approach," in *Proc. of the 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, 2008.

[14] K. Seyerlehner, T. Pohle, G. Widmer, and D. Schnitzer, "Informed selection of frames for music similarity computation," in *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, 2009.

[15] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, 2007.

[16] M. Levy and M. Sandler, "Lightweight measures for timbral similarity of musical audio," in *Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia*, 2006.

[17] G. Tzanetakis and P. Cook, "Musical genre classification of audio signal," *IEEE Transactions on Audio and Speech Processing*, 2002.