# FRAME LEVEL AUDIO SIMILARITY - A CODEBOOK APPROACH

*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 and
Austrian Research Institute for AI, Vienna
gerhard.widmer@jku.at

*Peter Knees*
Dept. of Computational Perception
Johannes Kepler University
Linz, Austria
peter.knees@jku.at

## ABSTRACT

Modeling audio signals by the long-term statistical distribution of their local spectral features - often denoted as bag of frames approach (BOF) - is a popular and powerful method to describe audio content. While modeling the distribution of local spectral features by semi-parametric distributions (e.g. Gaussian Mixture Models) has been studied intensively, we investigate a non-parametric variant based on vector quantization (VQ) in this paper. The essential advantage of the proposed VQ approach over state-of-the-art similarity measures is that the proposed audio similarity metric forms a normed vector space. This allows for more powerful search strategies, e.g. *KD-Trees* or *Local Sensitive Hashing* (LSH), making content-based audio similarity available for even larger music archives. Standard VQ approaches are known to be computationally very expensive; to counter this problem, we propose a multi-level clustering architecture. Additionally, we show that the multi-level vector quantization approach (ML-VQ), in contrast to standard VQ approaches, is comparable to state-of-the-art frame-level similarity measures in terms of quality. Another important finding w.r.t. the ML-VQ approach is that, in contrast to GMM models of songs, our approach does not seem to suffer from the recently discovered hub problem.

## 1. INTRODUCTION

The rapid growth of music material resulting from ever increasing data storage and transmission capabilities has motivated the research in new methods to manage and interact with large music archives. One of the basic building blocks to enable new ways of interaction with music collections (e.g. new ways of browsing and searching for audio material) is to automatically determine similarities among songs. To realize such a building block, there are basically two strategies. The first strategy is to automatically extract information about music by crawling the web or by analyzing user feedback with respect to a given song. The second strategy is to extract information directly out of the music signal itself. While web based music information retrieval is gaining in popularity because of the increasing amount of meta information available on the web, we believe there is still room for improvement on the audio modeling side. One such improvement is presented in this paper.

Ideally a content-based audio similarity metric should approximate the ill-defined "sounds like" relation for songs (e.g $Song_A$ "sounds like" $Song_B$). This is of course not a trivial task, especially since this relation depends on the individual perception of various musical aspects. So two songs can be perceived to be similar because of their instrumentation, rhythmic structure, singing voice, timbre, melody, tempo, lyrics and even because of common social backgrounds. A look at the algorithms which have been submitted to the Music Information Retrieval Evaluation eXchange (MIREX'07[1]) — a competitive evaluation of music information retrieval algorithms —, reveals that most algorithms contain a component which models the overall spectral characteristic of a song by a statistical distribution of local spectral features. The overall spectral characteristic is related to a song's timbre and seems to be a very important subcomponent of state-of-the-art content-based audio similarity measures. While the most popular variants to model the distribution of local spectral features are either just a single multidimensional Gaussian distribution [1] or a mixture of several Gaussian components [2], we propose to use a non-parametric distribution, based on a multi-level vector quantization (ML-VQ) approach. Our approach will, in contrast to the very strange similarity spaces resulting from single Gaussian models or mixture models, form a normed vector space. This can be very beneficial, if one has to search for elements in such a similarity space or if one wants to visualize a similarity space. Furthermore the similarity space does not seem to be affected by the *hub*-problem (see section 5) and the song models obtained by the ML-VQ approach are rather simple and intuitively interpretable, which definitely helps to analyze the models' content as we do in section 6, where we describe the reconstruction of a song's spectrum based on the codebook and where we also explain how to resynthesize these reconstructions.

In the following section we discuss advantages and disadvantages of GMMs, and give a short overview on related work concerning vector quantization. Section 3 describes the proposed ML-VQ approach in detail. In section 4 we present the results of eval-

[1]http://www.music-ir.org/mirex/2007/index.php

uation experiments, where the ML-VQ approach is compared to a state-of-the-art algorithm. Finally, section 7 concludes on the presented results and ideas and gives an outlook on future work.

## 2. RELATED WORK

The first approaches [3, 2] to model timbral similarity were based on Gaussian Mixture Models (GMMs), which were used to model the distribution of local spectral features. The most prominent local spectral features in use were Mel Frequency Cepstrum Coefficients (MFCCs). MFCCs are a compact representation of the spectral envelope of a short audio frame and were and are still one of the most widespread features in the MIR community. While the general approach is still conceptually the same for state-of-the-art algorithms, GMMs have some shortcomings. One of the major drawbacks is the time consuming training process, which relies on the Expectation Maximization (EM) algorithm. The second crucial shortcoming is that comparing two distributions modeled by a GMM is not trivial at all. The Kullback-Leibler (KL) divergence [4] or relative entropy is a measure of the difference between two probability distributions, $P$ and $Q$.

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx \tag{1}$$

However for GMMs there exists no closed form formula to compute the KL distance. Therefore the only way to compute the KL distance is to approximate the KL distance by Monte-Carlo sampling of MFCC vectors from one distribution and estimating the likelihood of these vectors given the other distribution, which of course is quite computationally expensive. Another popular variant to compute the distance of two distributions is the Earth Movers Distance (EMD)[5]. Unfortunately the EMD is quite expensive as well.

To speed up the overall process various simplified distribution models were under consideration (e.g. GMMs with diagonal covariance matrices only, or with a reduced number of components). Surprisingly, it turned out that a single multivariate Gaussian distribution of MFCCs can perform as well as mixture models [1, 6]. This did not only reduce the size of the models themselves, but also simplified the computation of the KL divergence a lot, since a closed form exists for the KL divergence for single Gaussians. Furthermore the training process simplifies to the computation of mean and covariances over MFCC vectors, altogether resulting in a dramatically improved performance of the similarity measure. Nevertheless, it is important to note that the similarity computation for an entire collection is still exhaustive, to be more precise will require $(N^2 - N)/2$ distance computations, because the KL distance does not fulfill the triangle inequality and therefore one cannot easily apply more powerful search strategies. Consequently, it is especially desirable to have a model which can be interpreted as a point in a vector space, because this will enable non-exhaustive search strategies like *KD-tree* or *Local Sensitive Hashing* (LSH). The recent findings of Aucouturier [7] further emphasize that the similarity space based on GMMs or single Gaussians is a quite strange space. There exist some songs in music collections that according to these similarity measures are similar to almost any song invariably the music collection, while there cannot be found any relevant perceptual similarity. These false positives are called *hubs* and seem to be related to the model or the distance computation according to [7, 8]. For a more detailed view on the hub problem we refer to section 5.

Vector Quantization, as a non-parametric density estimation method, has not received much research attention. Foote and Pye were to our knowledge the first to design a music genre classification system based on vector quantization [9, 10]. They use a supervised tree-based quantization schema to learn a decision tree from the test-set, which splits the MFCC feature space into maximally discriminative regions with respect to the associated genre. For each song of the training set they compute a histogram over the leaf nodes of the tree, after subdividing the MFCC vectors according to the trained tree structure. These histograms are then compared to genre histogram templates by using euclidean or cosine distance to predict the genre. On the one side the major strength of this approach is to efficiently generate a global quantization structure, but on the other side the approach is limited because of its supervised nature and the strong bias of tree learning algorithms. A more appropriate way to come up with a global segmentation of the feature space are unsupervised clustering algorithms, e.g. *k-means* or *self-organizing maps*. Self-Organizing Maps (SOMs) have been proposed by [11] and have been evaluated in [6]. The SOM-VQ approach, according to the results in [6], seems to perform worse than the single Gaussian or GMM variant. The *k-means* clustering algorithm has been investigated in [8] by Aucouturier. Additionally, he also investigates a supervised variant known as *Learning Vector Quantization* (LVQ). For both variants he reports classification results about 15% less precise than GMMs. However he makes two interesting observations. First of all he points out that finding a global codebook is a computationally very intense task, since all feature vectors of all songs have to be clustered at once to generate a global codebook. To reduce the computational costs, he proposes to subsample the overall distribution. Secondly, he observes that the quality increases with the number of songs used to create the codebook, whereas the number of frames per song appears not to be a crucial factor, and concludes that the frames of a single song might be quite redundant. These are two interesting observations, since the first observation clearly identifies the main disadvantage of the VQ approach, while the second observation already gives a hint on how to reduce the computational costs. In the next section we introduce our multi-level VQ approach, which tries to reduce the computational complexity of the VQ approach based on this observation.

## 3. A MULTILEVEL VECTOR QUANTIZATION APPROACH

Our vector quantization approach is based on Lloyd's variant [12] of the *k-means* clustering algorithm to partition the overall feature space into $k$ quantization regions. This variant of the *k-means* algorithm was chosen because Lloyd's iterative refinement heuristic is known to converge very quickly and is therefore a good choice, since we have to deal with a huge number of feature vectors. For this standard variant there is no guarantee on the quality of the resulting clustering. It depends heavily on the chosen initial vectors. Consequently, we decided to use a special seeding algorithm, proposed by Arthur and Vassilvitskii [13], known as the *k-means++* algorithm. They have shown that by applying the proposed seeding technique, the resulting clustering can be expected to be $\Omega(\log k)$ worse than the optimal clustering. Thus, seeding ensures some quality guarantees on our global codebook. Still to deal with large audio collections we have to reduce the number of feature vectors to come up with a global codebook in reasonable time.

### 3.1. Main Architecture

One way of reducing the number of feature vectors is to simply randomly subsample the overall distribution of feature vectors. We propose to make use of Aucouturier's observation that the quality of the codebook increases with the number of songs used to generate the codebook, rather than the number of frames per song [8]. Redundant feature vectors from a single song just increase computational costs, but do not improve the quality of the global partitioning of the feature space. Consequently it seems to be advantageous to already remove redundant feature vectors at the song-level. To do so, we use the *k-means++* algorithm to cluster the feature vectors within individual songs first and then pass these song-level cluster centers to the global codebook generation stage, where once again a *k-means++* algorithm is used to generate the final codebook. Figure 1 gives an overview on this process. This multi-level clustering architecture greatly reduces the computational costs, of course depending on both the number of cluster centers at the song-level and the number of cluster centers in the final codebook generation stage.

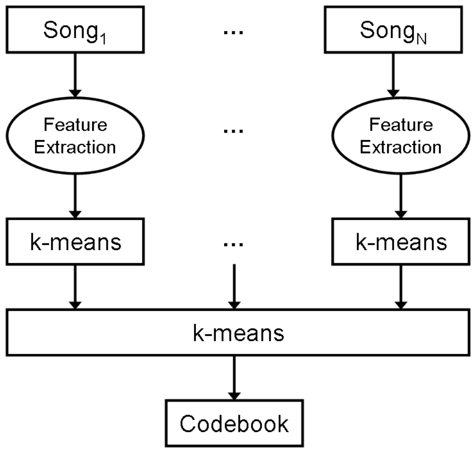

Figure 1: *Overview of the multi-level vector quantization (ML-VQ) approach.*

### 3.2. Feature Extraction

For feature extraction the input signal is converted to 11kHz mono. Then we perform a Short Time Fourier Transform (STFT) using a window size of 2048 samples, a hop size of 256 samples, a hanning window and take the power spectrum $|X(f)|^2$ thereof. To account for the musical nature of the content, we sum up all frequency bins within a constant bandwidth of 25 cents starting from 2050 cent (equal to about 53.43 Hz), which corresponds to a mapping onto a logarithmic musical scale [14]. The resulting spectral feature vectors still have 321 dimensions.

$$f_{cent} = 1200 \log_2(f_{Hz}/(440 \times 2^{\frac{3}{12}-5})) \tag{2}$$

This results in a linear frequency resolution up to about 1178 Hz and starts compressing the higher frequency content thereafter in a logarithmic way. Note that we have also evaluated a multirate filterbank implementation instead of the very simple summing across frequency bins to better approximate the musical scale in equation (2), but except dramatically increased computational costs no improvement in quality could be achieved. Finally, we transform this compressed power spectrum $X(k)$ according to equation (3) to obtain a logarithmic amplitude scale.

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

Additionally, to speed up the clustering at the song level, we do not cluster all audio frames, but instead select a fixed number of so-called *"event vectors"* for each song, quite similar to [15]. Event vectors should capture onsets and can be identified by an onset detection function. In our implementation a simple time domain power based onset detection function is used to identify onset frames. For each song only the $n$ most significant onset frames are kept and used during the song-level clustering. An additional speedup can be achieved if we use the time domain onset detection function to only transform those windows to the spectral domain which correspond to event vectors. Figure 2 illustrates how the onset detection function can be used to only transform onset windows to the frequency domain. Note that the feature extraction process, described in this subsection, is an essential part of two processes: the codebook generation process and the model generation process (see next subsection).

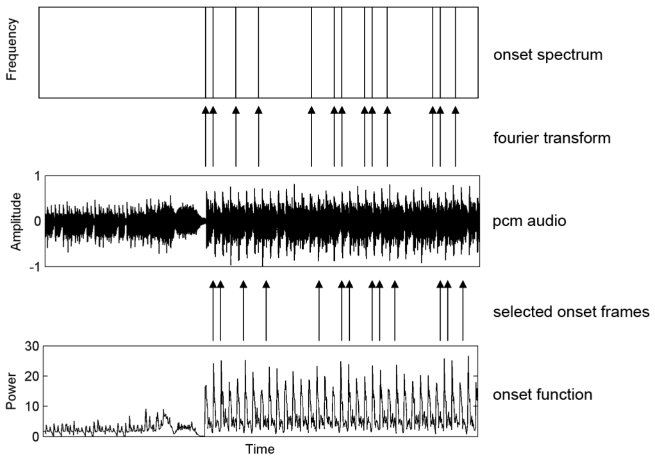

Figure 2: *The onset detection function is used to select those time domain windows, which have to be transformed to the frequency domain. The resulting spectral frames are so-called "event vectors".*

### 3.3. Song Model Generation

Once a general codebook has been constructed, song models based on this codebook have to be generated. To generate a histogram model of a song (either one of the songs involved in codebook generation, or a new one), we extract $n$ event vectors for this song as described in subsection 3.2. Then a histogram over the $m$ quantization units (cluster centers) of the codebook is built. Each event vector $E_j$ is mapped to its closest codebook vector $C_{u_j}$, where $u_j$ is the index of the closest codebook vector and is computed according to equation (4). For each codebook vector $C_i$ we end up with a corresponding histogram bin $H_i$ indicating the number of event vectors mapped to the $i$-th codebook vector, see equation

(5). To measure the distance between codebook vectors and event vectors the *City-Block* or $L_1$ distance is used.

$$u_j = \arg\min_{k\leq m} |E_j - C_k| \tag{4}$$

$$H_i = \sum_j^n u_j == i \tag{5}$$

### 3.4. Comparison

Various types of distance measures can be used to compare two histograms. In [10] euclidean and cosine distance are proposed. We decided to compute the *histogram intersection* as introduced by Swain and Ballard for image indexing [16]. Given a pair of histograms, $I$ and $M$, each containing N bins, the histogram intersection distance is defined by equation (6).

$$D(I,M) = 1 - \frac{\sum_{i=1}^m \min(I_i, M_i)}{\sum_{i=1}^m M_i} \tag{6}$$

In our special case, where the sum over the histogram bins is constant ($\sum_i^m I_i = \sum_i^m J_i = n$) the histogram intersection reduces to the the *City-Block* or $L_1$ distance.

$$D(I,M) = 1 - \frac{\sum_{i=1}^m \min(I_i, M_i)}{\sum_{i=1}^m M_i} = \frac{1}{2n}\sum_{j=1}^m |I_i - M_i| \tag{7}$$

Histogram intersection is a very simple and fast distance measure, which can even be computed incrementally [16]. It is important to note that the histogram intersection distance is a full featured metric implying non-negativity, identity of indiscernibles, subadditivity and symmetry. Consequently any histogram can be interpreted as a point in a normed vector space, which probably allows to apply more powerful search algorithms e.g. Local Sensitive Hashing (LSH). This could be especially useful in the context of very large audio archives and is an essential advantage over the KL divergence for example, which does not fulfill the triangle inequality and is not symmetric by itself.

## 4. CLASSIFICATION EXPERIMENTS

### 4.1. Dataset

For lack of reliable ground truth w.r.t perceived audio similarity, we follow the standard procedure in MIR research and evaluate our similarity measure in an indirect way, via music genre classification. We generated a genre classification dataset of freely available songs from *download.com*[2]. To ensure reasonable track quality only the 190 most popular songs according to the number of total listens for each genre were taken. Although some of the songs had to be removed[3], the number of songs per genre is almost equal. Altogether there are 3180 tracks from 1517 different artists distributed over 19 genres in this dataset. Table 1 gives an overview on the genres and their precise distribution. Compared to other evaluations datasets, this dataset is quite large, has an almost equal genre distribution and contains tracks from a high number of different artists.

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

[3]The crawled genre information in our current MATLAB implementation is stored in the id3 tags of the songs, which for some reason could not be handled by the external library for some of the songs.

| **genre** | **#tracks** | **#artists** |
|---|---|---|
| Blues | 186 | 100 |
| Country | 187 | 103 |
| Hip-Hop | 155 | 87 |
| Jazz | 177 | 103 |
| New Age | 175 | 82 |
| Reggae | 172 | 83 |
| Classical | 125 | 46 |
| Folk | 185 | 98 |
| Latin | 163 | 86 |
| Rock & Pop | 181 | 117 |
| Alternative & Punk | 182 | 116 |
| Electronic & Dance | 164 | 92 |
| R&B & Soul | 175 | 113 |
| World | 158 | 76 |
| Religious | 172 | 71 |
| Children's | 164 | 74 |
| Easy Listening & Vocals | 175 | 98 |
| Comedy & Spoken Word | 134 | 68 |
| Soundtracks & More | 150 | 72 |
| **total** | **3180** | **1517** |

Table 1: *The track and artist distribution over 19 genres of the dataset used in our evaluations. Note that the number of artists of the whole collection is not equal to the sum over the individual genres, since some songs of one and the same artist belong to different genres.*

### 4.2. Evaluation Procedure and Results

In our evaluation the multi-level VQ (**ML-VQ**) approach is compared to the single Gaussian (**SG**) component of the similarity algorithm proposed by Pohle and Schnitzer [17], which took the first rank in the MIREX 2007 Audio Music Similarity and Retrieval competition. Furthermore we also present results for a random algorithm (**RND**) to represent the baseline. To give a comprehensive overview of the quality, we compute different quality indicators. All these indicators are related to a query scenario, where the algorithm is asked to return a set of songs that *sound like* the query song. For such a result set one counts the number of *correctly* returned songs. A song in the result set is assumed to be correct w. r. t. our evaluation, if the genre is the same as the genre of the query song, due to the lack of more precise ground truth information. The following paragraph gives a mathematically more precise definition of what kind of quality measures are evaluated.

Consider a music collection of $n$ tracks separated into $p$ genres, then a classification function *classify*, which counts the number of songs belonging to genre $g$ in a result set of size $r$ given $S_i$ as query song, is defined by

$$\text{classify}(S_i, g, r) = \sum_{j=1}^n \mathcal{G}(S_j) == g \wedge \mathcal{R}(S_j|S_i) < r \tag{8}$$

where $\mathcal{G}(S_j)$ denotes the genre of song $S_j$ and $\mathcal{R}(S_j|S_i)$ denotes the rank according to the similarity measure of the song $S_j$ given the song $S_i$ as query song.

- **k-NN accuracy** (k-NN Acc)
  The $k$-nearest neighbor (k-NN accuracy) classification ac-

curacy for a collection of size $n$ is given by

$$Acc_k = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} \text{classify}(S_i, \mathcal{G}(S_i), k)}{kn}. \quad (9)$$

The k-NN accuracy is an important measure, because in quite many applications only the top ranked songs, hence the most similar songs, are of use. In our evaluation we present results for the 1-NN accuracy and 5-NN accuracy.

- **Average per class classification accuracy** (k-NN AvgAcc) The average per class classification accuracy is the mean over the k-NN classification accuracies of individual genres. The k-NN classification accuracy of a specific genre $G_h$ of size $n_h$ is defined by

$$Acc_k(G_h) = \frac{\sum_{i=1}^{n} \text{classify}(S_i, G_h, k)}{kn_h} \quad (10)$$

The average per class classification accuracy is then defined as

$$AvgAcc_k = \frac{\sum_{i=1}^{p} Acc_k(G_i)}{p} \quad (11)$$

In contrast to the total $k$-NN classification accuracy, where high classification results on individual genres can lead to a high overall classification accuracy, the average per class k-NN accuracy provides a more reliable estimate of the expected classification accuracy on individual genres.

- **R-precision** (R-Prec) The R-precision in contrast to k-NN measures considers all songs of the genre of the query song. Hence, for a given query song the size of the result set is equivalent to the size of the song's genre. The R-precision for a query song $S_i$ is defined as

$$Prec(S_i) = \frac{\text{classify}(S_i, \mathcal{G}(S_i), n_{\mathcal{G}(S_i)})}{n_{\mathcal{G}(S_i)}} \quad (12)$$

The global R-precision over a whole collection is defined as the mean over all songs.

$$Prec = \frac{\sum_{i=1}^{n} prec(S_i)}{n} \quad (13)$$

- **Artist Filter** (AF) An artist filter removes tracks from the same artist as the query song $S_i$ from the dataset before any evaluations are performed. Algorithms capturing artist-specific song properties will likely yield higher accuracy values on nearest neighbor-based quality indicators. Our interest is of course to find interesting similarity relations between songs of different artists and not trivial similarity relations between songs of one and the same artist. Therefore we also apply an artist filter and present evaluation results for both variants, with and without artist filter.

In table 2 all the results from these evaluations are summarized. The ML-VQ approach seems to perform sightly better when an artist filter is in use. The single Gaussian model seems to outperform the ML-VQ approach when all relevant documents are considered (R-precision), but altogether we conclude that both approaches perform equally well in terms of quality.

| AF | indicator | ML-VQ | SG | RND |
|---|---|---|---|---|
| without | 1-NN Acc | 36.65% | **37.33%** | 5.03% |
| | 1-NN AvgAcc | 37.16% | **37.70%** | 5.04% |
| | 5-NN Acc | 28.16% | **28.22%** | 5.63% |
| | 5-NN AvgAcc | **28.67%** | 28.56% | 5.56% |
| | R-Prec | 0.138 | **0.141** | 0.061 |
| with | 1-NN Acc | **22.83%** | 21.57% | 4.97% |
| | 1-NN AvgAcc | **23.07%** | 21.85% | 4.98% |
| | 5-NN Acc | **19.96%** | 18.70% | 5.58% |
| | 5-NN AvgAcc | **20.17%** | 18.86% | 5.51% |
| | R-Prec | 0.136 | **0.140** | 0.050 |

Table 2: *Evaluation results for three similarity algorithms: multi-level vector quantization (**ML-VQ**), single Gaussian (**SG**), random guess (**RND**); **k-NN Acc** = k-nearest-neighbor accuracy, **k-NN AvgAcc** = Average per class classification accuracy, **R-prec** = R-precision (see subsection 4.2).*

## 5. THE HUB PROBLEM

Recent research findings indicate that the similarity spaces based on GMMs or single Gaussians are rather strange spaces. There exist songs in music collections that according to these classes of algorithms are similar to almost any other song, while no perceptual relevant similarity can be found. In accordance with Aucouturier [7, 2] we define that a song is a so-called *hub* song,

- if the song appears among the nearest neighbors of most songs in the database

**and**

- if most of these appearances do not correspond to any meaningful perceptual similarity.

To quantify the *"hubness"* of a song Aucouturier introduced the $n$-Occurrence of a song as a natural measure of hubness. The $n$-Occurrence is the number of times a song occurs in the first $n$ nearest neighbors of all the other songs in the dataset. Taking a look at the distribution of the songs in a dataset according to the number of their $n$-Occurrences, it turns out that most songs have a very low $n$-Occurrence and just a few songs have a very high $n$-Occurrence — these are the hubs.

Aucouturier observed that the numbers of songs with a given $n$-Occurrence follow an exponential distribution. We could also observe this type of distribution on our dataset for the single Gaussian approach (see figure 3). However, the distribution of $n$-Occurrences of the ML-VQ approach (visible in figure 4) differs significantly. Even more importantly, the plots of the $n$-Occurrences themselves (upper plots in Figs. 3 and 4) are clearly different. Comparing these two plots for SG and ML-VQ it immediately jumps to the eye that there are no songs for the ML-VQ approach which have comparably extreme n-Occurrence values as for the SG approach. Since extreme $n$-Occurrence values are an indicator for hubs, we conclude that our approach is apparently *hub-free*, which actually contradicts the results obtained in [8].

## 6. FROM CODEBOOKS TO SONG RECONSTRUCTION

Any vector quantization algorithm partitions the input or feature space into disjoint regions, and this partitioning determines the quality of the quantization. In some cases – e.g., with tree learners

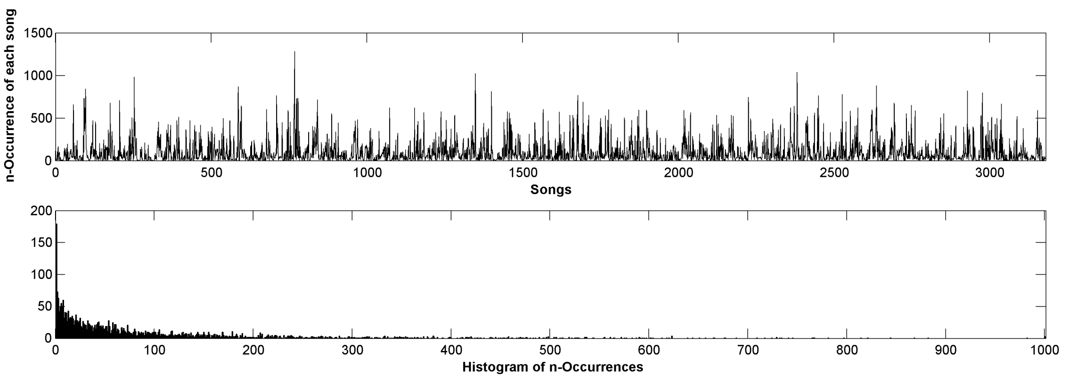

Figure 3: *Visualization of the 100-Occurrences of the* ***single Gaussian*** *distance measure (upper plot) and the corresponding histogram, showing the expected exponential decay. Extreme 100-Occurrence values (upper plot) indicate hubs.*

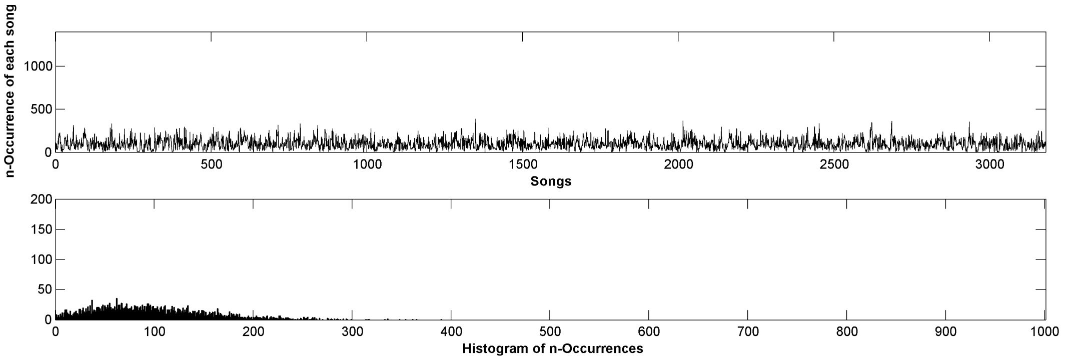

Figure 4: *Visualization of the 100-Occurrences of the* ***histogram intersection*** *distance measure (upper plot) and the corresponding histogram. Obviously the distribution is quite different compared to the distribution in figure 3. Most notably there are no songs, which have an extreme 100-Occurrence value (upper plot), which implies that there are no hubs.*

– this partitioning has an explicit representation. In clustering-based vector quantization, the partitions are implicitly defined by the most prototypical vector in each region – the so-called codebook vectors. A very interesting question is surely if there are methods to improve our codebooks. To that end, it would be helpful to understand what is actually represented by our current codebook. This could probably allow us to conclude on what is not captured and on how to improve the codebook.

One important advantage of our VQ approach is that the resulting models are very intuitive to interpret. To get an auditive impression of what is captured by a song's model, we simply have to transform a song's spectrum into its codebook representation. To do so, we extract all feature frames $F_j$ (see subsection 3.2) of a song, not just the event frames, and map each frame to its most similar codebook vector $C_i$ according to equation (14).

$$u_j = \arg \min_{k \in \mathcal{N} \leq m} |F_j - C_k| \tag{14}$$

Then we replace the original feature vector $F_j$ by the corresponding codebook vector $C_{u_j}$. Finally, to come up with the reconstructed *model spectrum* we have to invert the logarithmic transformation of the amplitude scale, uncompress the high frequencies according to equation (2) and take the square root to transform the power into the magnitude spectrum.

Knowing what is actually captured by a model (attributed to the reconstructed spectrum), we are also interested in what kind of information is lost. So we compute the residual by subtracting the reconstructed model spectrum $X_m$ from the original magnitude spectrum $X_o$.

$$X_r = X_o - X_m \tag{15}$$

The residual spectrum $X_r$ consists of two different *"components"*. There is the *positive residual spectrum* $X_r^+$ and the *negative residual spectrum* $X_r^-$ which are defined by

$$X_r^+ = \begin{cases} X_r & X_r \geq 0 \\ 0 & X_r < 0 \end{cases} \tag{16}$$

and

$$X_r^- = \begin{cases} |X_r| & X_r \leq 0 \\ 0 & X_r > 0. \end{cases} \tag{17}$$

These two residual signals result from the fact that the reconstructed model spectrum sometimes underestimates the original spectrum, resulting in a positive residual, but also overestimates the original spectrum, resulting in a negative residual. We believe that it is important to treat these two residual signals separately, since the negative residual is essentially noise introduced by the model, whereas the positive residual corresponds to peaks in the original spectrum that could not be approximated by the model and carries some quite useful information.

Once we have obtained these magnitude spectra, it is quite straightforward to resynthesize a song. We perform a block-by-block synthesis by computing the *inverse Fast Fourier Transform* (iFFT) for each frame, overlap the short audio chunks and add them up, as for example described in [18]. Since no phase information is captured by the codebook elements, which would not make much sense, we use the phase of the original spectrum. Of course the phase information is not part of the model, but as we resynthesize both the reconstructed spectrum as well as the residual spectra the audible difference of these signals can only result from the change of the magnitude spectrum, since the phase information stays the same in both cases. Another interesting property of this decomposition of the original signal $s_o$ into the model signal $s_m$, the positive residual signal $s_r^+$ and the negative residual signal $s_r^-$ is their additivity.

$$s_o = s_m + s_r^+ - s_r^- \tag{18}$$

Thus, one can for example easily subtract from the model signal, the noise signal introduced by the model, which helps in analyzing the resynthesized signals.

By listening to the separated audio components we found that quite little information is captured by our model. This implies that the similarity judgment of a state-of-the-art frame-level audio similarity metric is based on little and very reduced information of the analyzed song. Somewhat surprisingly we also found that our VQ approach tends to capture percussive elements more precisely than tonal components. This is an interesting finding and might be related to our feature extraction process. As pointed out in section 3.2 one strategy to reduce the computational effort of frame clustering at the song-level is to only cluster *event vectors*. Because of the chosen onset function, which is used to identify event frames, it is very likely that event vectors preferably capture percussive elements. In some way we unintentionally restricted the algorithm to model songs preferably by transient events. An example of such a signal or spectrum decomposition is visualized in figure 5 and some audio examples can be found on the web[4].

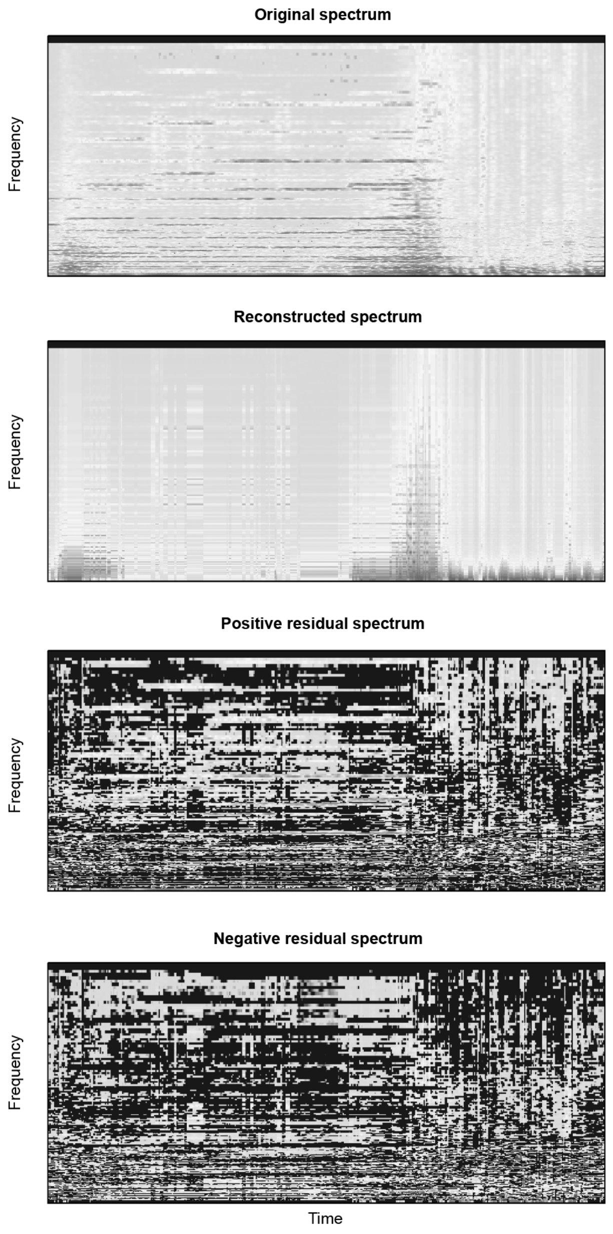

Figure 5: *Signal decomposition into original, reconstructed, positive and negative magnitude spectrum in (dB). The original signal is only very roughly approximated by the codebook. Percussive elements starting at the end of the audio clip are far better approximated than the sinusoidal components in the first two-thirds of this audio chunk.*

[4]www.cp.jku.at/people/seyerlehner/vq/vq.html

## 7. DISCUSSION, CONCLUSION AND FUTURE WORK

This paper has presented a new algorithm for modeling audio content based on local spectral features, using multi-level vector quantization. The proposed multi-level VQ approach performs comparably to state-of-the-art frame-level audio similarity algorithms, which we have shown by a broad experimental evaluation. However, compared to state-of-the-art approaches, the ML-VQ approach has considerable advantages. Most notably the proposed histogram intersection distance implies a normed vector space. This allows to apply more powerful search strategies (e.g. *KD-Tree* or *Local Sensitive Hashing*) enabling content-based music retrieval for even larger music archives. We could also show that the resulting audio similarity space is *hub-free*. Furthermore the model itself is rather simple and intuitively interpretable, which led to the idea to represent a song's spectrum in terms of codebook vectors. This revealed that rather little information is stored in the current song models. In particular, our approach seems to focus on percussive elements. This of course raises new questions. Can we create even better codebooks? Can we generate codebooks, which focus on other aspects than just percussive events? Hence, can we for example create a piano specific codebook? Another interesting idea is to represent a song only by the song-level codebook vectors of another song or perhaps manually specify which codebook elements should be replaced by other codebook elements. The result could be some automatic or semi-automatic content-based audio effect.

Last but not least, some future research could focus on how one could maintain a constantly adapting codebook. This will be important because in the current approach the codebook is trained once and for all and does not allow to adapt to changes within a collection, e.g songs which are being added or removed.

## 8. ACKNOWLEDGMENTS

This research was supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under grants L112-N04 and L511-N15.

## 9. REFERENCES

[1] M. Mandel and D. Ellis, "Song-level features and svms for music classification," in *In Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR'05*, London, UK, 11-15th September 2005.

[2] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high's the sky?," *J. Negative Results Speech Audio Sci.*, vol. 1, no. 1, 2004.

[3] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *In proceedings IEEE International Conference on Multimedia and Expo (ICME)*, Tokyo, Japan, August 2001.

[4] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

[5] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *In Proceedings of the 1998 IEEE International Conference on Computer Vision, ICCV'98*, Bombay, India, January 1998, pp. 59–66.

[6] M. Levy and M. Sandler, "Lightweight measures for timbral similarity of musical audio," in *AMCMM '06: Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, Santa Barbara, California, USA, 2006, pp. 27–36.

[7] J.-J. Aucouturier and F. Pachet, "A scale-free distribution of false positives for a large class of audio similarity measures," *Pattern Recogn.*, vol. 41, no. 1, pp. 272–284, 2008.

[8] J.-J. Aucouturier, *Ten experiments on the modelling of polyphonic timbre*, Ph.D. thesis, University of Paris 6, 2006.

[9] D. Pye, "Content-based methods for the management of digital music," in *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, Washington, DC, USA, Sept. 17-23, 2006, pp. 2437–2440.

[10] J. T. Foote, "Content-based retrieval of music and audio," in *In Multimedia Storage and Archiving Systems II, Proc. of SPIE*, 1997, pp. 138–147.

[11] F. Vignoli and S. Pauws, "A music retrieval system based on user-driven similarity and its evaluation," in *In Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR'05*, London, UK, 11-15th September 2005.

[12] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.

[13] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, Louisiana, 2007, pp. 1027–1035.

[14] M. Goto, "Smartmusickiosk: Music listening station with chorus-search function," in *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology, UIST'03*, November 2003, pp. 31–40.

[15] C. Yang, "The macsis acoustic indexing framework for music retrieval: An experimental study," in *In Proceedings of the 3rd International Conference on Music Information Retrieval, ISMIR'02*, Paris, France, 2002.

[16] M.J. Swain and D.H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, November 1991.

[17] T. Pohle and D. Schnitzer, "Striving for an improved audio similarity measure," in *Third Music Information Retrieval Evaluation eXchange, MIREX'07*, Vienna, Austria, 2007.

[18] U. Zölzer, *DAFX - Digital Audio Effects*, WILEY, Southern Gate, Chichester, England, 2002.