INSTRUMENTATION-BASED MUSIC SIMILARITY USING SPARSE REPRESENTATIONS

Hiromasa Fujihara[⋆†] *Anssi Klapuri*[⋆] *Mark D. Plumbley*[⋆]

[⋆] Queen Mary University of London, Centre for Digital Music
[†] National Institute of Advanced Industrial Science and Technology (AIST)

ABSTRACT

This paper describes a novel music similarity calculation method that is based on the instrumentation of music pieces. The approach taken here is based on the idea that sparse representations of musical audio signals are a rich source of information regarding the elements that constitute the observed spectra. We propose a method to extract feature vectors based on sparse representations and use these to calculate a similarity measure between songs. To train a dictionary for sparse representations from a large amount of training data, a novel dictionary-initialization method based on agglomerative clustering is proposed. An objective evaluation shows that the new features improve the performance of similarity calculation compared to the standard mel-frequency cepstral coefficients features.

Index Terms— Music similarity, Instrumentation, Sparse representation, Online dictionary learning

1. INTRODUCTION

This paper describes a method for measuring the similarity of music pieces based on their instrumentation. The term *instrumentation* refers to the particular combination of instruments employed in a piece of music and the way in which the music is arranged for the instruments. Our method can be used for content-based music retrieval, allowing users to search for songs that have similar instrumentation. Together with conventional bibliographic information such as song and artist names, we believe that the proposed method enables more efficient music retrieval.

Content-based music information retrieval (MIR) is rapidly gaining in importance as the number of songs users can access has exploded and portable audio players and online music stores have become widely prevalent. The trend is likely to continue, along with the popularization of network-connected smartphones with music player functionality and large-capacity cloud storage services.

Reflecting the above developments, there have been a number of studies on content-based MIR [1], and MIREX [2], an annual evaluation of MIR algorithms, has been held since 2005 with an increasing number of participants. While many of the studies use relatively low-level features such as mel-frequency cepstral coefficients (MFCCs), spectral centroid, or frame-to-frame spectral change, some studies have proposed feature extraction techniques to describe specific elements of music such as singing voice [3] and instrumentation [4, 5].

To develop instrumentation-based MIR system, feature vectors that represent instrumentation have to be extracted from polyphonic music. Kitahara et al. [4] and Pei et al. [5] learned models of pre-defined instrument classes in advance by using training data with annotations. However, considering the number of instruments appearing in music, it is not realistic to prepare training data and learn models for all the instruments conceivable, and therefore their method has a limitation in terms of versatility.

We propose a novel feature extraction approach based on *sparse representations* [6] that does not require supervised learning. In a sparse representation of music, a spectrum is decomposed into a weighted sum of *dictionary* elements (a set of bases). The term *sparse* refers to a constraint that only a small number of elements in the dictionary are active at each time. This method is suitable for analyzing musical audio signals where the number of concurrent instruments and notes playing at each time is quite limited compared to the set of possible instruments and musical sounds which is very large. For example, a full piano keyboard consists of 88 keys, but generally no more than 10 keys are pressed at any given time. The dictionary used for sparse decomposition can be learned from training data in a unsupervised manner, meaning that data with annotations are not required. The learned dictionary is expected to consist of spectra of instruments in the training data, and, hence, sparse representations calculated by using the dictionary can be considered as a representation of the instrumentation of music. Although Sturm et al. [7] proposed a similarity search system for speech signal using sparse representation based on a pre-defined Gabor dictionary, they did not utilize dictionary learning.

In the rest of this paper, boldface lower and upper case symbols will be used to denote vectors and matrices, respectively. We use $||\boldsymbol{x}||_1$ and $||\boldsymbol{x}||_2$ to denote the ℓ_1 and ℓ_2-norm of a vector $\boldsymbol{x}$, respectively, so that $||\boldsymbol{x}||_2 = \sqrt{\sum_i x_i^2}$ and $||\boldsymbol{x}||_1 = \sum_i |x_i|$.

2. MUSIC SIMILARITY BASED ON SPARSE REPRESENTATION

We propose a representation of musical audio signals based on sparse representation that can be used to calculate similarities of songs in terms of instrumentation. Assuming that a *dictionary* (a set of bases) $\boldsymbol{A}$ is known, sparse coding decomposes observed magnitude spectra $\boldsymbol{X}$ into weighted sums of dictionary elements according to

$$\boldsymbol{X} = \boldsymbol{AS} + \boldsymbol{E}, \tag{1}$$

where the *activation matrix* $\boldsymbol{S}$ encodes the activations of the dictionary elements and $\boldsymbol{E}$ is an error matrix representing additive noise. Here additivity of magnitude spectra is assumed and $\boldsymbol{A}$ and $\boldsymbol{S}$ are constrained to non-negative values (thus the model is an instance of non-negative sparse coding [8]). Our motivation for the above model stems from the idea that the activation matrix contains rich information on the composition of the observed spectra and therefore we expect them to be good feature vectors for calculating similarities based on instrumentation. The dictionary $\boldsymbol{A}$ is learned from training data in an unsupervised manner in advance.

One of the most important features of non-negative sparse coding is that the activation matrix $\boldsymbol{S}$ is allowed to have only a small number of non-zero elements by introducing a sparsity constraint when estimating $\boldsymbol{S}$. Without the sparsity constraint, the problem is underdetermined when the dictionary is overcomplete, that is, when the number of bases exceeds the number of frequency bins in the spectra. This means that the number of dictionary elements is limited to the number of frequency bins. The sparsity constraint allows us to use an overcomplete dictionary by implementing a trade-off between the number of active dictionary elements and the representation error compared to the observed spectrum. Since we expect the activation matrix to represent the instrumentation of songs, the dictionary has to consist of a large number of bases that can cover the

This work was supported by EU Framework 7 FET-Open project FP7-ICT-225913-SMALL: Sparse Models, Algorithms and Learning for Large-Scale data and a Leadership Fellowship (EP/G007144/1) from the UK Engineering and Physical Sciences Research Council (EPSRC).

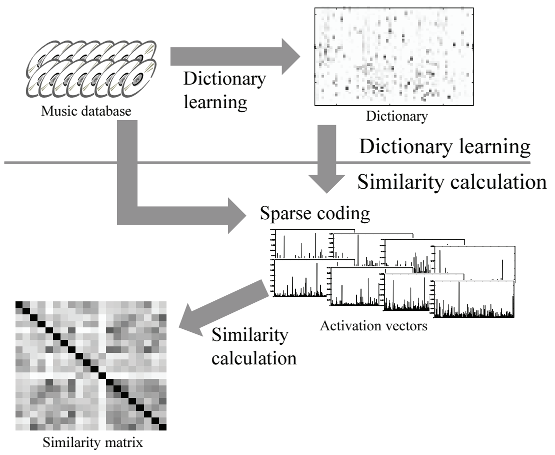

Fig. 1. Ovewview of our method.

sounds appearing in various genres of music.

The formulation of non-negative matrix factorization (NMF) [9, 10, 11] is very similar to Eq. (1), except that NMF usually estimates the dictionary and the activation matrix jointly from observed spectra. However, although NMF (and other similar algorithms) has flourished in many domains of music information research including automatic transcription [9], genre classification [10], and music tagging [11], to the authors' knowledge, there have been no studies which trained a basis matrix from a large number of songs and applied it for calculating music similarity.

Figure 1 shows an overview of our method including the feature extraction and the calculation of music similarity based on instrumentation. First, a dictionary is learned from a music database. Note that the dictionary learning is performed in an unsupervised manner, requiring no annotations. Note, also, that the database used for training dictionary can be either different from, or the same as, that for which the similarities are to be calculated. Then, each song in the music database is decomposed into an activation matrix by performing sparse coding with the trained dictionary. Finally, by considering the activation matrices as feature vectors, we can calculate similarities between songs.

2.1. Non-negative sparse coding

In this section, we elaborate the formulation of non-negative sparse coding and the estimation of the activation matrix. An input audio signals is converted to a magnitude spectrogram $H(w,t)$ by using the short-term Fourier transform (STFT), where ω and t denote frequency and time indexes, respectively. The spectrogram $H(\omega,t)$ is normalized according to

$$X(w,t) = \frac{|H(\omega,t)|}{\sqrt{\sum_{\omega'}|H(\omega',t)|^2}} \tag{2}$$

where $X(\omega,t)$ is a normalized magnitude spectrogram. The normalization is essential to make a parameter of sparsity constraint consistent (as explained later in this section).

Sparse coding is based on a linear generative model, which represents a spectrum as an instantaneous mixture of dictionary elements. Given that

$$\boldsymbol{x}(t) = (X(1,t),\cdots,X(\omega,t),\cdots,X(\Omega,t))^T \tag{3}$$

is a spectrum with Ω frequency bins at time t, we decompose this using the model

$$\boldsymbol{x}(t)=\boldsymbol{A}\boldsymbol{s}(t)+\boldsymbol{e}(t) \tag{4}$$

$$=\sum_k \boldsymbol{a}_k S_k(t)+\boldsymbol{e}(t) \tag{5}$$

where $\boldsymbol{e}(t)$ is a noise vector,

$$\boldsymbol{A} = (\boldsymbol{a}_1,\cdots,\boldsymbol{a}_k,\cdots,\boldsymbol{a}_K) \tag{6}$$

represents a dictionary matrix with K elements, and

$$\boldsymbol{s}(t) = (S_1(t),\cdots,S_k(t),\cdots,S_K(t))^T \tag{7}$$

denotes the activation vector at time t.

Assuming that $\boldsymbol{e}(t)$ is a zero-mean Gaussian random vector with a constant diagonal covariance matrix $\boldsymbol{I}$, the conditional probability density function $p(\boldsymbol{x}|\boldsymbol{s};A)$ can be written as

$$p(\boldsymbol{x}|\boldsymbol{s};A) \propto \exp\left\{-\frac{||\boldsymbol{x}-A\boldsymbol{s}||_2^2}{2}\right\}. \tag{8}$$

A zero-mean Laplace distribution with a scale parameter $\frac{1}{\lambda}$ is used as a prior for $\boldsymbol{s}$,

$$p(\boldsymbol{s};\lambda) \propto \exp\{-\lambda||\boldsymbol{s}||_1\}. \tag{9}$$

Then, maximum a posteriori (MAP) estimate of $\boldsymbol{s}$ is given by

$$\hat{\boldsymbol{s}}=\underset{\boldsymbol{s}}{\operatorname{argmax}}\{\log p(\boldsymbol{x}|\boldsymbol{s};\boldsymbol{A})+\log p(\boldsymbol{s};\lambda)\} \tag{10}$$

$$=\underset{\boldsymbol{s}}{\operatorname{argmax}}\left\{-\frac{1}{2}||\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s}||_2^2-\lambda||\boldsymbol{s}||_1\right\}. \tag{11}$$

This problem is called *lasso* [12].

Above, λ plays an important role because it makes the activation vector sparse, i.e. the larger the lambda is, the smaller the number of atoms that will be used to represent a spectrum. Since Eq. (11) implies that the value of λ depends on the scale of $\boldsymbol{x}$, the normalization of $\boldsymbol{x}$ as shown in Eq. (2) is introduced in an attempt to alleviate this dependency.

2.2. Dictionary learning

A dictionary used for the sparse coding here should consist of elements that are likely to appear in many music genres. We try to obtain such a dictionary by learning it from a large amount of training data. Since the dictionary learning is unsupervised, the target data for which similarities are to be calculated can also be included in the training data without loss of generality.

Several algorithms have been proposed to learn a dictionary for sparse coding [13, 14]. An online algorithm is required in our case because of the large amount of data used for the learning: it is not realistic to store spectrograms of a few hundred songs in memory. Therefore we use the online algorithm for dictionary learning proposed by Mairal et al. [14].

Given N training samples $\{\boldsymbol{x}(1),\cdots,\boldsymbol{x}(n),\cdots,\boldsymbol{x}(N)\}$, the objective function to be optimized [14] is given by

$$L = \underset{\boldsymbol{A},\boldsymbol{S}}{\operatorname{argmin}}\sum_{n=1}^{N}\frac{1}{2}||\boldsymbol{x}(n)-\boldsymbol{A}\boldsymbol{s}(n)||_2^2+\lambda||\boldsymbol{s}(n)||_1. \tag{12}$$

The basic strategy of the dictionary learning algorithm is to update $\boldsymbol{A}$ and $\boldsymbol{S}$ one at a time. $\boldsymbol{S}$ can be optimized by the LARS algorithm that will be explained later.. Optimization of $\boldsymbol{A}$ can be done using an algorithm based on block-coordinate descent proposed by Mairal et al. [14]. The entire algorithm for dictionary learning is summarized in Algorithm 1.

To solve the lasso problem several approaches have been proposed (a good summary can be found in [14]). In this paper we adopt LARS [15], which is used in [14] because of its advantage in terms of a computational time when the activation vector $\boldsymbol{s}$ is very sparse.

2.3. Clustering method for dictionary initialization

The choice of the initial dictionary used in Algorithm 1 is important because the dictionary updating algorithm is based on block-coordinate descent, which is a kind of hill-climbing technique. This may lead to a poor local minimum if the initial dictionary is inappropriate. We prepare the initial dictionary using the following steps;

1. A small dictionary with K_I elements is learned for each song in the training data by using Algorithm 1. Random elements selected from the training song is used as an initial dictionary in this step. In the implementation of this paper, we set $K_I = 20$.
2. All the small dictionaries are concatenated to obtain a large dictionary with $K_I \times N$ elements.

Algorithm 1 Dictionary learning algorithm [14].

```
Initialize $\boldsymbol{A} = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k, \cdots, \boldsymbol{a}_K)$ as explained in Section 2.3.
$\boldsymbol{B}_0 \leftarrow \boldsymbol{0}, \boldsymbol{C}_0 \leftarrow \boldsymbol{0}$.
for $n = 1 \rightarrow N$ do
    Estimate $\boldsymbol{s}(n)$ from $\boldsymbol{x}(n)$ and $\boldsymbol{A}$ by using LARS algorithm.
    $\boldsymbol{B}_n \leftarrow \boldsymbol{B}_{n-1} + \boldsymbol{s}(n)\boldsymbol{s}(n)^T$.
    $\boldsymbol{C}_n \leftarrow \boldsymbol{C}_{n-1} + \boldsymbol{x}(n)\boldsymbol{s}(n)^T$.
    repeat
        for $k = 1 \rightarrow K$ do
            Update the $k$-th column $a_k$ of the dictionary $\boldsymbol{A}$.
            $\boldsymbol{u}_k \leftarrow \frac{1}{\boldsymbol{B}_{n(k,k)}}(\boldsymbol{c}_k - \boldsymbol{A}\boldsymbol{b}_k)$.
            $\boldsymbol{a}_k \leftarrow \frac{1}{\max(||\boldsymbol{u}_k||_2, 1)}\boldsymbol{u}_k$.
        end for
    until convergence
end for
```

3. The dictionary size is reduced to K by using an agglomerative clustering algorithm based on Euclidean distance[16].
4. Finally, starting from the obtained initial dictionary with K elements, a dictionary is trained by Algorithm 1 with the entire training data.

2.4. Similarity calculation

Finally, similarities between songs are calculated by considering the activation matrices $\boldsymbol{S}$ obtained by sparse decomposition as sequences of feature vectors. We calculate the mean activation vector for each song and define similarity as the cosine distance between the mean activations of different songs.

3. EVALUATION

Simulation experiments were carried out to objectively evaluate the proposed similarity calculation method by comparing the performance of our method and the conventional MFCC-based similarity caluculation method. Instead of conducting subjective evaluation using human listeners, we first define a groud-truth similarity matrix, which is created based on manually annotated instrument labels, and then calculate the distance between similarity matricies.

3.1. Conditions

We use a music database consisting of 504 songs taken from commercial CD recordings[1]. They are mono, and sampled at 16-bits resolution and 44.1 kHz rate. The genres of the songs vary from classical to rock and pop and from dance to jazz music. The database was originally created for the purpose of music classification in general and the balance between genres was made according to an informal estimate of what people listen to. The songs have instrument annotations made by a research assistants who wrote down the instruments from album covers and then verified them by listening to each piece. Since the original annotations included somewhat ambiguous labels such as "Sax." and "Saxophone", we reclassified these labels into 73 instrument classes by hand. To create a ground truth similarity matrix, we first represented each song by a 73-dimension vector, where each element corresponds to an instrument and assumes a value 0 or 1 depending on whether the song employs the respective instrument. The similarity matrix of size 504×504 was then calculated based on the cosine distance between the vectors. The audio signals of the songs were downsampled to 16kHz and converted to the spectrograms using a 1024-sample (64ms) STFT and the Hanning window. The hop-size of the STFT was set to 1600 samples (100ms).

[1]This database is the same as the one used in [17] and the songs are listed in `http://www.cs.tut.fi/sgn/arg/klap/musicDB.html`.

Table 1. Experimental results: the baseline methods. The labels (i), (ii), and (iii) are explained in Section 3.1.

(i) Random	(ii) MFCC	(iii) MFCC+Timbral
0.415	0.444	0.422

Table 2. Experimental results: the proposed methods. The values in parentheses represent unbiased standard deviations.

λ	$K = 500$	$K = 1000$	$K = 1500$	$K = 2000$
0.025	0.469 (0.001)	0.471 (0.001)	0.471 (0.001)	0.473 (0.002)
0.05	0.471 (0.001)	0.472 (0.001)	0.473 (0.001)	0.473 (0.002)
0.1	0.471 (0.001)	0.473 (0.001)	0.474 (0.001)	**0.475** (0.001)
0.2	0.471 (0.001)	0.473 (0.001)	0.472 (0.002)	**0.475** (0.001)
0.4	0.469 (0.002)	0.468 (0.001)	0.471 (0.001)	0.470 (0.001)

To calculate a distance between the ground truth similarity matrix and the estimated similarity matrix, we use the measure proposed in [18], which is based on practice in text information retrieval [19]. This measure treats each row of the similarity matrices as the results of a query for the corresponding song. For each row, i, the top R songs in the ground truth matrix are considered. Provided that the rth rank song in the ground truth matrix appears at the $k_{r,i}$th rank in the reference matrix, the *score* of the row is defined as $s_i = \sum_{r=1}^{R} \alpha_r^{r-1} \alpha_c^{k_{r,i}-1}$, where α_r and α_c are parameters, and i represents a row index in the matrix. After calculating the score for each row of the matrix, the total score is defined as $S = \frac{1}{I} \sum_{i=1}^{I} \frac{s_i}{s_{max}}$, where s_{max} represents the best possible value of the score calculated by $s_{max} = \sum_{r=1}^{R} \alpha_r^{r-1} \alpha_c^{r-1}$. We set $R = 10, \alpha_r = 0.5^{\frac{1}{3}}$, and $\alpha_c = \alpha_r^2$ as suggested in [18].

As a baseline, we use the following three methods:
(i) Random : A random similarity matrix, each element of which is randomly set between 0 to 1.
(ii) MFCC : Mel-frequency cepstral coefficients (MFCCs) are used as features and a similarity matrix is calculated based on the cosine distance.
(iii) MFCC+Timbral : Zero crossing rate, spectral centroid, flux and roll-off are appended with MFCCs to the feature vector and a similarity matrix is calculated based on the cosine distance.
The features in (ii) and (iii) are extracted using the MARSYAS 0.4.2 toolbox [20] with `-timbral` and `-mfcc` options, respectively, and applying the `-sv` option that causes MARSYAS to extract one feature vector per song. MARSYAS first extracts feature vectors by using 512 sample frames with no overlap, then calculates running means and standard deviations using 40 frame window, and finally calculates a mean and standard deviation over the entire song.

3.2. Results

We evaluated our method using λ values 0.025, 0.05, 0.1, 0.2, and 0.4, and dictionary sizes K of 500, 1000, 1500, and 2000. In the experiments, we changed random seeds for selecting initial dictionary elements of small dictionaries in 5 different values and calculate means and unbiased standard deviations. Tables 1 and 2 show results for the baseline methods and for the proposed method, respectively. As can be seen, the proposed method outperforms the baseline methods for all values of λ and K tested. The optimum score 0.475 is achieved with λ is 0.1 or 0.2 and K is 2000, although the values of λ and K have only a negligible influence on the score. Compared to the baseline method, the proposed method improved 0.060 points from (i) Random and 0.031 points from (ii) MFCC. Concerning the value of K, a weak positive correlation can be observed between K value and the scores. As regards the computational time, it took approximately 10 hours to learn dictionary and 40 minutes to calculate the similarity matrix of the 504 songs when λ is 0.1 and K is 2000.

Figure 2 shows examples of feature vectors that were obtained by averaging the activation vectors over time. In Figure 2, (a) and (b) represent classical music played by orchestras, whereas (c) and

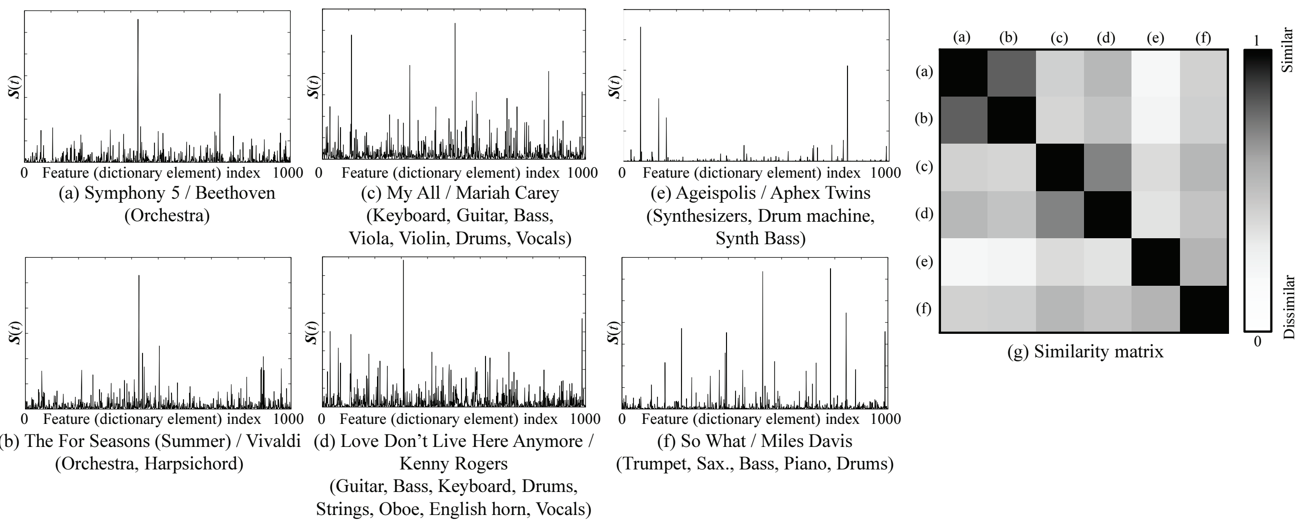

Fig. 2. Examples of feature vectors and a similarity matrix. Title, artist or composer name, and instrument annotation are shown underneath the figures. It can be seen that the distance between (a) and (b) is small, both of which are composed of similar instrumentation. The same applies to (c) and (d).

(d) represent popular music with similar instruments such as guitar, bass, drums, keyboard, and vocals. As can be seen from the similarity matrix (g), the similarites within each pair was relatively small for these two pairs of songs. The other songs shown in Fig. 2 were taken from various genres. We can see that pairs of songs that have similar instrumentation, namely (a) and (b) as well as (c) and (d), have feature vectors of similar shape, while the instrument labels and the shape of the vectors are different from each other among the songs of the different genres.

4. CONCLUSIONS

We have described a novel feature extraction method for measuring the similarity of instrumentation in different pieces of music. The underlying idea is to use sparse coding to decompose a mixture spectrum into a weighted sum of dictionary elements, and to use the vector of weights (the sparse activations) as a feature vector. The dictionary used in the non-negative sparse coding is learned from a large amount of music data in an unsupervised manner.

Our method extracts a single feature vector from a song by computing a time average over the activation matrix. This approach is straight-forward and easy to incorporate with other types of music recommendation systems such as collaborative filtering. On the other hand, our method can also extract a sequence of feature vectors from a song by omitting the time-averaging. Several methods have been proposed to calculate the similarity between two sequences of feature vectors, such as earth mover's distance and cross-likelihood ratio test. We plan to evaluate the proposed representation with such similarity calculation methods.

5. REFERENCES

[1] E. Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*, Ph.D. thesis, Universitat Wien, 2006.

[2] J. S. Downie, "The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research," *Acoust. Sci. Technol.*, vol. 28, pp. 247–255, 2008.

[3] H. Fujihara, M. Goto, T. Kitahara, and H. G. Okuno, "A modeling of singing voice robust to accompaniment sounds and its application to singer identification and vocal-timbre-similarity based music information retrieval," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, pp. 638–648, 2010.

[4] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Musical instrument recognizer "instrogram" and its application to music retrieval based on instrumentation similarity," in *Proc. ISM*, 2006, pp. 265–272.

[5] S.-C. Pei and N.-T. Hsu, "A novel music similarity measure system based on instrumentation analysis," in *Proc. ICME*, 2009, pp. 470–473.

[6] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse representations in audio and music: from coding to source separation," *Proc. IEEE*, vol. 98, pp. 995–1005, 2010.

[7] B. L. Sturm and L. Daudet, "On similarity search in audio signals using adaptive sparse approximations," in *Proc. AMR*, 2009, pp. 59–71.

[8] P. O. Hoyer, "Non-negative sparse coding," in *Neural Netw. Signal Process. XII (Proc. NNSP 2002)*, 2002, pp. 557–565.

[9] P. Smaragdis and J. C. Brown, "Non-negative matrix factorizatio for polyphonic music transcription," in *Proc. WASPAA*, 2003.

[10] A. Holzapfel and Y. Stylianou, "Musical genre classification using nonnegative matrix factorization-based features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, pp. 424–434, 2008.

[11] Y. Panagakis, C. Kotropoulos, and G. R. Arce, "Sparse multi-label linear embedding within nonnegative tensor factorization applied to music tagging," in *Proc. ISMIR*, 2010, pp. 393–398.

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[13] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, pp. 4311–4322, 2006.

[14] J. Mairal, *Sparse Coding for Machine Learning, Image Processing and Computer Vision*, Ph.D. thesis, Ecole Normale Superieure de Cachan, 2010.

[15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–499, 2004.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley, 2nd edition, 2001.

[17] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, pp. 342–355, 2006.

[18] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman, "A large-scale evaluation of acoustic and subjective music similarity measures," *Comput. Music J.*, vol. 28, no. 2, pp. 63–76, 2004.

[19] J. S. Breese, D. Heckerman, , and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. UAI*, 1998, pp. 43–52.

[20] G. Tzanetakis and P. Cook, "MARSYAS: A framework for audio analysis," *Organised Sound*, vol. 4, no. 30, pp. 169–175, 1999.