SEGMENTATION, CLUSTERING, AND DISPLAY IN A PERSONAL AUDIO DATABASE FOR MUSICIANS *

Guangyu Xia Dawen Liang Roger B. Dannenberg Mark J. Harvilla
Carnegie Mellon University
{gxia, dawenl, rbd, mharvill}@andrew.cmu.edu

*Published in: *Proceedings of the 12th International Society for Music Information Retrieval Conference,* October 2011, Miami, Florida.

ABSTRACT

Managing music audio databases for practicing musicians presents new and interesting challenges. We describe a systematic investigation to provide useful capabilities to musicians both in rehearsal and when practicing alone. Our goal is to allow musicians to automatically record, organize, and retrieve rehearsal (and other) audio to facilitate review and practice (for example, playing along with difficult passages). We introduce a novel music classification system based on Eigenmusic and Adaboost to separate rehearsal recordings into segments, an unsupervised clustering and alignment process to organize segments, and a digital music display interface that provides both graphical input and output in terms of conventional music notation.

1. INTRODUCTION

Music Information Retrieval promises new capabilities and new applications in the domain of music. Consider a personal music database composed of rehearsal recordings. Music is captured by continuously recording a series of rehearsals, where the music is often played in fragments and may be played by different subsets of the full ensemble. These recordings can become a valuable resource for musicians, but accessing and organizing recordings by hand is time consuming.

To make rehearsal recordings more useful, there are three main processing tasks that can be automated. (See Figure 1.) The first is to separate the sound into music and non-music segments. The music segments will consist of many repetitions of the same material. Many if not most of the segments will be fragments of an entire composition. We want to organize the segments, clustering them by composition, and aligning them to one another (and possibly to other recordings of the music). Finally, we want to coordinate the clustered and aligned music with an interface to allow convenient access.

We see these capabilities as the foundation for an integrated system in which musicians can practice and compare their intonation, tempo, and phrasing to existing recordings or to rehearsal data from others. By performing alignment in real time, the display could also turn pages automatically.

The next section presents a novel method for music/non-music classification and segmentation. Section 3 describes how to organize the segments. Section 4 describes a two-way interface to the audio.

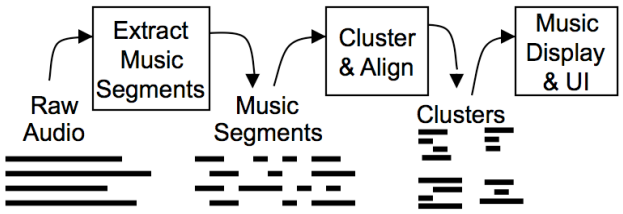

Figure 1. System diagram for a musician's personal audio database. Rehearsal recordings are automatically processed for simple search, analysis, and playback using a music notation-based user interface.

2. CLASSIFICATION AND SEGMENTATION

2.1 Related Work

Much work has been done in the area of classification and segmentation on speech and music. For different tasks, people extract different features. Some focus on background music detection [6], while others detect speech or music sections in TV programs or broadcast radio. Many features have been tested in the realm of speech/music classification [8, 17]. Two frequently used ones are Spectral-Centroid and Zero-Crossing Rate. Also, different statistical models have been used. Two of them, long window sampling [7] and the HMM segmentation framework [1, 14], are especially relevant to our work. Other approaches include using decision trees [16] and Bayesian networks [5].

However, the particular problem of variations in the sound source seems to be largely ignored. In reality, sound is not standardized in volume or bandwidth and may even contain different kinds of noise. In these cases, more robust features and methods are needed. This section will concentrate on new feature extraction and model design methods

to achieve music/non-music classification and segmentation on realistic rehearsal audio.

2.2 Eigenmusic Feature Extraction

The concept of Eigenmusic is derived from the well-known representation of images in terms of Eigenfaces [12]. The process of generating Eigenmusic can be performed in both the time and frequency domains, and in either case, simply refers to the result of the application of Principal Component Analysis (PCA) to the audio data [3]. Therefore, Eigenmusic refers to the eigenvectors of an empirical covariance matrix associated with an array of music data. The array of music data is structured as a spectrogram and hence contains the spectral information of the audio in those time intervals. When expressing non-music data in terms of Eigenmusic, the coefficients are generally expected to be outlying based on the fundamentally different characteristics of music and non-music.

In practice, we use about 2.5 hours of pure music in the training data collection to extract the Eigenmusic in the frequency domain. First, let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]$ be a spectrogram, a matrix consisting of, in its columns, magnitude spectra corresponding to 1.25 second non-overlapping windows of the incoming music data. Second, the corresponding empirical covariance matrix, $\boldsymbol{C}_x$, and its Eigenvectors are computed. Ultimately, we retain the first 10 eigenvectors corresponding to the largest eigenvalues. If $\boldsymbol{P}$ is the matrix of column-wise eigenvectors of $\boldsymbol{C}_x$, given a new magnitude spectrum column vector $\boldsymbol{x}$, we can represent its Eigenmusic coefficients by $\boldsymbol{P}^T\boldsymbol{x}$, which will be a 10-dimensional vector.

2.3 Adaboost Classifier

Adaboost [18] is a very interesting classification algorithm, which follows a simple idea: to develop a sequence of hypotheses for classification and combine the classification results to make the final decision. Each simple hypothesis is individually considered a *weak classifier*, $h(\boldsymbol{P}^T\boldsymbol{x})$, and the combined complex hypothesis is considered to be the *strong classifier*. In the training step, each weak classifier focuses on instances where the previous classifier failed. Then it will obtain a weight, α_t, and update the weight of individual training data based on its performance. In the decoding step, the strong classifier is taken to be the sign of the weighted sum of weak classifiers:

$$H(\boldsymbol{x}) = sign(\sum_t \alpha_t h_t(\boldsymbol{P}^T\boldsymbol{x})) \quad (1)$$

By training a sequence of linear classifiers h_t, each one of which merely compares an individual Eigenmusic coefficient against a threshold that minimizes the weighted error, Adaboost is able to implement a non-linear classification surface in the 10-dimensional Eigenmusic space.

2.3.1 Data Collection and Representation

The Adaboost training data is a collection of about 5 hours of rehearsal and performance recordings of western music; while the testing data is a collection of 2.5 hours of Chinese music. For the music parts, each data collection contains different combinations of wind instruments, string instruments, and singing. For the non-music parts, each data collection contains speech, silence, applause, noise, etc. Both data collections are labeled as music or non-music at the frame level (1.25 seconds). From Section 2.2, we know that each frame is a point in the 10-dimensional Eigenmusic space. Therefore, we have about 5 (hours) × 3600 (s/hour) / 1.25 (s/frame) = 14,400 frames for training and 7,200 frames for testing.

2.3.2 Implementation and Evaluation

We train 100 weak classifiers to construct the final strong classifier. The testing accuracy is shown in Figure 2. The results were obtained in terms of the percentage of error at the frame level. Two different statistics have been calculated: the percentage of true music identified as non-music, shown as the solid line, and the percentage of true non-music identified as music, shown as the dotted line.

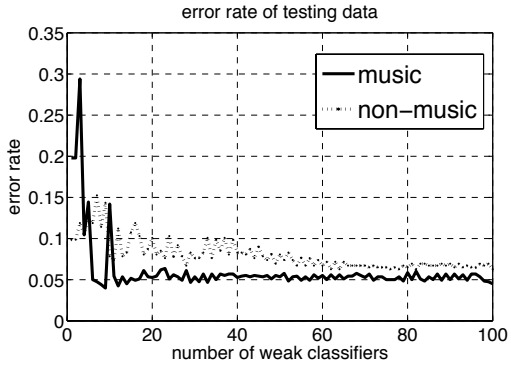

Figure 2. The testing error of music and non-music.

From Figure 2, it can be seen that the proposed Adaboost classifier in the Eigenmusic space is capable of achieving a low error rate (about 5.5%) on both music and non-music data, even when the testing data comes from a completely different sound source from the training data.

2.3.3 Probabilistic Interpretation

We can improve the frame level classification by considering that state changes between music and non-music do not occur rapidly. We can model rehearsals as a two-state hidden Markov model (HMM) [13]. Formally, given a vector $\boldsymbol{x}$, let $y \in \{-1,1\}$ represent its true label. Here, -1 stands for non-music and 1 stands for music. And let $w(\boldsymbol{x})$ represent the weighted sum of weak classifiers:

$$w(\boldsymbol{x}) = \sum_t \alpha_t h_t(\boldsymbol{P}^T \boldsymbol{x}) \quad (2)$$

In Equation (1), we took the sign of $w(\boldsymbol{x})$ as the decision, but we can modify this approach to compute the *a posteriori* probability of $y = 1$, given the weighted sum, which we denote as the function F:

$$F(w(\boldsymbol{x})) = P(y = 1 \mid w(\boldsymbol{x})) \quad (3)$$

According to the discussion in [15], $F(w(\boldsymbol{x}))$ is a logistic function, as shown in Equation 4:

$$F(w(\boldsymbol{x})) = \frac{1}{1 + \exp(-2 \cdot w(\boldsymbol{x}))} \quad (4)$$

In Figure 3, the small circles show $P(y = 1 \mid w(\boldsymbol{x}))$ estimated from training data sorted into bins according to $w(\boldsymbol{x})$. The logistic function is shown as the solid curve. It can be seen that our empirical data matches the theoretical probability quite well.

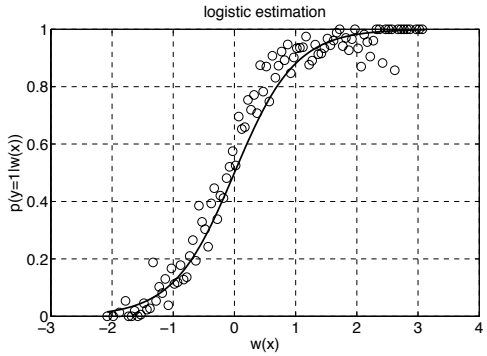

Figure 3. The logistic function estimation on training data.

We note that the idea of linking Adaboost with HMMs is not new, but very little work has been done to implement it [4, 19]. As far as we know, this is the first attempt of a probabilistic interpretation of Adaboost when linked with HMMs.

2.4 HMM Smoothing for Segmentation

The significance of smoothing is that even a very low error rate at the frame level cannot guarantee a satisfying segmentation result overall (i.e. at the piece level). For example, suppose a relatively low 5% error rate is obtained at the frame level. If the segmentation rule is to separate the target audio at every non-music frame, a 10 minute long pure music piece would be cut into about 25 pieces in this case. Ultimately, this is an undesirable result.

Based on typical characteristics of rehearsal audio data, we assume that: (1) music and non-music frames cannot alternate frequently, and (2) short duration music and non-music intervals are less likely than longer ones. By utilizing these assumptions in conjunction with the HMM, low (but possibly deleterious) frame-level error rates can be further reduced. We use a fully-connected HMM with only two states, representing music and non-music. The HMM observation corresponding to every frame $\boldsymbol{x}$ is a real number $w(\boldsymbol{x})$, as in Equation (2), given by the Adaboost classifier.

2.4.1 HMM Training

The training data collection mentioned in Section 2.3.1 is used to estimate the HMM parameters. Formally, let $\boldsymbol{S} = [S_1, S_2, \ldots, S_T]$ be the state sequence and let $\boldsymbol{O} = [O_1, O_2, \ldots, O_T]$ be the observation sequence. Since it is a supervised learning problem, we do Maximum Likelihood Estimation (MLE) by counting or just manually setting the parameters for initial state probabilities and transition probabilities. For emission probabilities, we use Bayes' rule:

$$P(O_t \mid S_t = 1) = \frac{P(S_t = 1 \mid O_t) \cdot P(O_t)}{P(S_t = 1)} \quad (5)$$

Remember that in our model $O_t = w(\boldsymbol{x}_t)$ and $P(O_t)$ is a constant. Therefore, if we plug in function F according to Equation (3), we obtain the estimate of the emission probability of music where C denotes a constant scalar multiplier:

$$P(O_t \mid S_t = 1) = C \cdot \frac{F(w(\boldsymbol{x}_t))}{P(S_t = 1)} \quad (6)$$

Using the same method, we obtain the estimate of the emission probability of non-music:

$$P(O_t \mid S_t = -1) = C \cdot \frac{1 - F(w(\boldsymbol{x}_t))}{P(S_t = -1)} \quad (7)$$

Here, we set the *a priori* probability of both music and non-music to 0.5 and then apply the Viterbi algorithm [13] to efficiently find the best possible state sequence for a given observation sequence.

2.4.2 Implementation and Evaluation

At the frame level, HMM smoothing reduced the error rate from about 5.5% to 1.8% on music and to 2.2% on non-music. This is the same as the best claimed result [17] in the references [6, 7, 8, 17], where classifiers were tested on cleaner data sets not related to our application. Since the piece level evaluation has been largely ignored in previous works on music/non-music segmentation, we adopt an evaluation method from speech segmentation [20] called Fuzzy Recall and Precision. This method pays more attention to insertion and deletion than boundary precision. We get a Fuzzy Precision of 89.5% and Fuzzy Recall of 97%. The high Fuzzy Recall reflects that all true boundaries are well detected with only some imprecision around the boundaries. The lower Fuzzy Precision reflects that about 10% of the detected boundaries are not true ones.

3. CLUSTERING OF MUSIC SEGMENTS

Assuming perfect classification results from the previous step, the clustering task is a distinct problem. Our goal is to cluster the musical segments belonging to the same piece.

3.1 Feature Extraction

Chroma vectors [2] have been widely used as a robust harmonic feature in all kinds of MIR tasks. The chroma vector represents the spectral energy distribution in each of the 12 pitch classes (C, C#, D,… A#, B). Such features strongly correlate to the harmonic progression of the audio.

Considering the objective that our system should be robust to external factors (e.g. audience cheering and applause), the feature cannot be too sensitive to minor variations. Therefore, as suggested by Müller, we first calculate 12-dimensional chroma vectors using 200ms windows with 50% overlap, then compute a longer-term summary by windowing over 41 consecutive short-term vectors and normalizing, with a 10-vector (1s) hop-size. These long-term feature vectors are described as *CENS features* (**C**hroma **E**nergy distribution **N**ormalized **S**tatistics) [10, 11]. The length of the long-term window and hop size can be changed to take global tempo differences into account.

3.2 Audio Matching and Clustering

Given the CENS features, audio matching can be achieved by simply correlating the query clip $\boldsymbol{Q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots \boldsymbol{q}_M)$ with the subsequences of musical segments $\boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots \boldsymbol{p}_N)$ in the database (assume $N > M$). Here, all lower case letters (e.g. $\boldsymbol{q}_i$, $\boldsymbol{p}_i$) represent 12-dimensional CENS vectors. Thus, $\boldsymbol{Q}$ and $\boldsymbol{P}$ are both sequences of CENS vectors over time. As in [11], the distance between the query clip $\boldsymbol{Q}$ and the subsequence $\boldsymbol{P}^{(i)} = (\boldsymbol{p}_i, \boldsymbol{p}_{i+1}, \ldots \boldsymbol{p}_{i+M-1})$ is:

$$dist(\boldsymbol{Q}, \boldsymbol{P}^{(i)}) = 1 - \frac{1}{M}\sum_{k=1}^{M} \langle \boldsymbol{q}_k, \boldsymbol{p}_{i+k-1} \rangle \qquad (8)$$

Here $\langle \boldsymbol{q}_k, \boldsymbol{p}_{i+k-1} \rangle$ denotes the dot product between these two CENS vectors. All of the distances for $i = 1, 2, \ldots N-M+1$ together can be considered a distance function Δ between query clip $\boldsymbol{Q}$ and each of the musical segments $\boldsymbol{P}$ in the database. If the minimum distance is less than a preset threshold γ, then $\boldsymbol{Q}$ can be clustered with $\boldsymbol{P}$.

One problem with this decision scheme is that, unlike a traditional song retrieval system which has a large reference database in advance, our system has no prior information about the rehearsal audio stream. We are only given a stream of potentially unordered and unlabeled audio that needs to be clustered. To solve this problem, we construct the database from the input audio dynamically. The inputs are all the music segments obtained from Section 2, and the algorithm is:

1. Sort all the music segments according to their length.
2. Take out the longest segment $\boldsymbol{S}$.
   i) If database $\boldsymbol{D}$ is empty, put $\boldsymbol{S}$ into $\boldsymbol{D}$ as a cluster.
   ii) Otherwise match $\boldsymbol{S}$ with every segment in $\boldsymbol{D}$ by calculating distance function Δ. Let $\boldsymbol{D}_\mathbf{m}$ be the segment in $\boldsymbol{D}$ with the best match.
      (1) If the distance function Δ of $\boldsymbol{D}_\mathbf{m}$ with $\boldsymbol{S}$ has a minimum less than γ, cluster $\boldsymbol{S}$ with $\boldsymbol{D}_m$.
      (2) Otherwise make $\boldsymbol{S}$ a new cluster in $\boldsymbol{D}$.
   iii) Repeat step 2 until all segments are clustered.

Here we made a critical assumption: the longest segment is most likely to be a whole piece or at least the longest segment for this distinct piece, so it is reasonable to let it represent a new cluster. At every step of the iteration, we take out a new segment $\boldsymbol{S}$ which is guaranteed to be shorter than any of the segments in database $\boldsymbol{D}$. This implies it can either be part of an existing piece in the database (in which case we will cluster it with a matching segment) or it is a segment for a new piece which does not yet exist in the database (in which case we will make it a new cluster).

We also need to consider the possibility that tempo differences cause misalignment between sequences. We can obtain different versions of CENS features (for example, from 10% slower to 10% faster) for the same segment to represent the possible tempos. This is achieved by adjusting the length of the long-term window and the hop size as mentioned in Section 3.1. During matching, the version of the segment with the lowest distance function minimum will be chosen.

3.2.1 Segment Length vs. Threshold Value

While time scaling compensates for global tempo differences, it does not account for local variation within segments. It is interesting to consider the length of the query clip that is used to correlate with the segments in the database. Intuitively, longer clips will be more selective, reducing spurious matches. However, if the length is too large, e.g. two segments both longer than 5 minutes, sequence misalignments due to tempo variation will decrease the correlation and increase the distance. If longer segments lead to greater distance, one might compensate with larger threshold values (γ). However, larger γ values may not prove strict enough to filter out noise, leading to clustering errors. We will compare two pairs of configurations: longer segments with larger γ and shorter segments with smaller γ.

3.2.2 Experiments and Evaluation

We have two parameters to control: γ, which determines if the two segments are close enough to be clustered together, and t, the length of the segments. We use hours of rehearsal recordings as test data, with styles that include classical,

rock, and jazz. We also use live performance recordings, which are typically even longer. To evaluate the clustering results, we use the F-measure as discussed in [9]:

$$P = \frac{TP}{TP + FP}, \; R = \frac{TP}{TP + FN} \tag{9}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{10}$$

Here, P (precision) and R (recall) are determined by 4 different variables: TP (true positive) which corresponds to assigning two similar segments to the same cluster, TN (true negative) corresponding to assigning two dissimilar segments to the different clusters, FP (false positive) corresponding to assigning two dissimilar segments to the same cluster, and FN (false negative) which corresponds to assigning two similar segments to different clusters. β is the tuning parameter used to adjust the emphasis on precision or recall. In our case, it is more important to avoid clustering segments from different pieces into one cluster than it is to avoid "oversegmenting" by creating too many clusters. The latter case is more easily rectified manually. Thus, we would like to penalize more on false positives, which leads to choosing $\beta < 1$. Here, we use $\beta = 0.9$. Considering the possible noise near the beginning and the end of the recordings, we choose the middle t seconds if the segment is shorter than the original recording.

As seen in Figure 4, for segments longer than 3 minutes, the relatively larger $\gamma = 0.25$ outperforms others, while for shorter segments around 20s to 60s, the smaller $\gamma = 0.15$ has the best performance. It is also shown that if γ is set too large (0.35), the performance drops drastically. Overall, shorter segments and smaller γ give us better results than longer segments and larger γ. Finally, since calculating correlation has $O(n^2)$ complexity, shorter segment lengths can also save significant computation. Thus, our current system uses a segment length $t = 40$s and $\gamma = 0.15$. *K-means* clustering was also tested but did not work as well as our algorithm because of the non-uniform segment length and unknown number of clusters (details omitted for reasons of space).

4. USER INTERFACE

Ultimately, we plan to integrate our rehearsal audio into a digital music display and practice support system (see Figure 5.). While listening to a performance, the user can tap on music locations to establish a correspondence between music audio and music notation. Once the music has been annotated in this manner, audio-to-audio alignment (a by-product of clustering) can be used to align other audio automatically. The user can then point to a music passage in order to call up a menu of matching audio sorted by date, length, tempo, or other attributes. The user can then practice with the recording in order to work on tempo, phrasing, or intonation, or the user might simply review a recent rehearsal, checking on known trouble spots. One of the exciting elements of this interface is that we can make useful audio available quickly through a natural, intuitive interface (music notation). It is easy to import scanned images of notation into the system and create these interfaces.

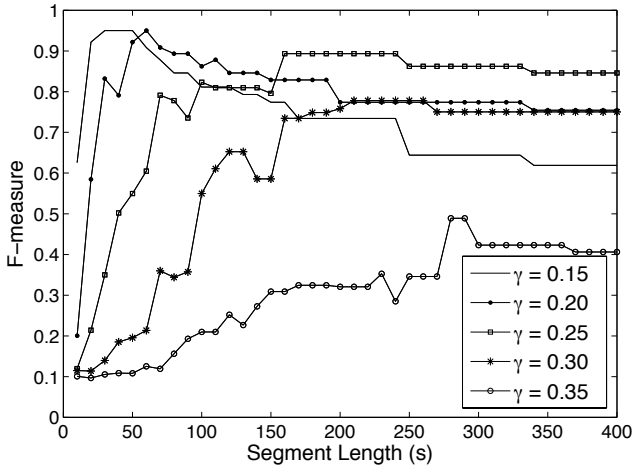

Figure 4. Experimental results with different segments of length t and matching threshold γ.

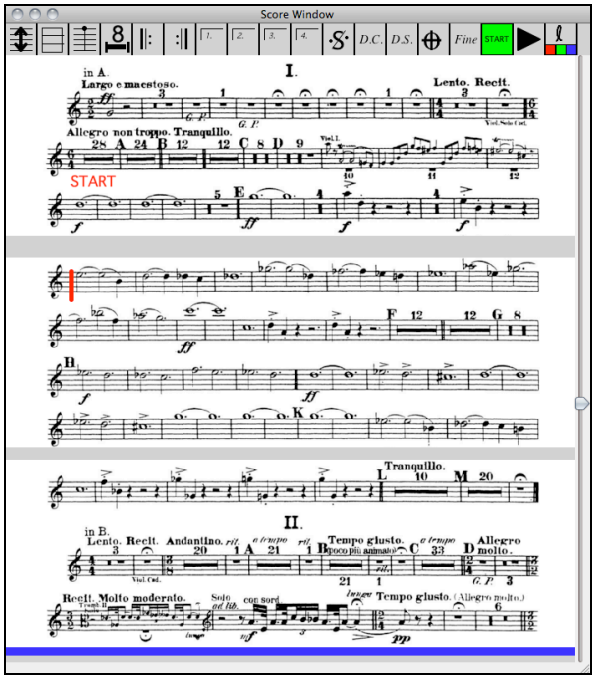

Figure 5. Audio database is accessed through a common music notation interface. The user has selected the beginning of system 3 as a starting point for audio playback, and the current audio playback location is shown by the thick vertical bar at the beginning of system 4.

5. CONCLUSIONS

We have presented a system for automated management of a personal audio database for practicing musicians. The system segments recordings and organizes them through unsupervised clustering and alignment. An interface based on common music notation allows the user to quickly retrieve music audio for practice or review. Our work introduces Eigenmusic as a music detection feature, a probabilistic connection between Adaboost and HMMs, an unsupervised clustering algorithm for music audio organization, and a notation-based interface that takes advantage of audio-to-audio alignment. In the future, we will fully integrate these components and test them with actual users.

6. ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Grant No. 0855958. We wish to thank Bhiksha Raj for suggestions and comments on this work, and the Chinese Music Institute of Peking University for providing recordings of rehearsal for analysis.

7. REFERENCES

[1] J. Ajmera, I. McCowan and H. Bourlard: "Speech/Music Segmentation Using Entropy and Dynamism Features in a HMM Classification Framework," *Speech Communi-cation* 40 (3), pp. 351-363, 2003.

[2] M. Bartsch and G. Wakefield: "To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 15-18, 2001.

[3] D. Beyerbach, H. Nawab: "Principal Components Analysis of Short-Time Fourier Transform," *International Conference on Acoustics, Speech, and Signal Processing*, 1991.

[4] C. Dimitrakakis and S. Bengio: "Boosting HMMs with an Application to Speech Recognition," *International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, 2004.

[5] T. Giannakopoulos, A. Pikrakis and S. Theodoridis: "A Speech/Music Discriminator for Radio Recordings Using Bayesian Networks," *International Conference on Acoustics, Speech, and Signal Processing*, 2006.

[6] T. Izumitani, R. Mukai, and K. Kashino: "A Background Music Detection Method Based on Robust Feature Extraction," *International Conference on Acoustics, Speech, and Signal Processing*, 2008.

[7] K. Lee and D. Ellis: "Detecting Music in Ambient Audio by Long-Window Autocorrelation," *International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, USA, 2008.

[8] G. Lu and T. Hankinson: "A Technique Towards Automatic Audio Classification and Retrieval," *Proceedings of ICSP*, Beijing, China, 1998.

[9] C. D. Manning, P. Raghavan, and H. Schütze: *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[10] M. Müller, S. Ewert, and S. Kreuzer: "Making Chroma Features More Robust to Timbre Changes," *International Conference on Acoustics, Speech, and Signal Processing*, pp. 1869-1872, Taipei, Taiwan, 2009.

[11] M. Müller, F. Kurth, and M. Clausen: "Audio Matching via Chroma-Based Statistical Features," in *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 288-295, 2005.

[12] D. Pissarenko: "Eigenface-based facial recognition," 2002.

[13] L. Rabiner: "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 77(2), pp. 257-287, 1989.

[14] C. Rhodes, M. Casey, S. Abdallah, and M. Sandler: "A Markov-chain Monte-Carlo approach to musical audio segmentation," *International Conference on Acoustics, Speech, and Signal Processing*, 2006.

[15] J. Riedman, T. Hastie, and R. Tibshirani: "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol 208, No.2, pp. 337-407, 2000.

[16] A. Samouelian, J. Robert-Ribes, and M. Plumpe: "Speech, Silence, Music and Noise Classification of TV Broadcast Material," *Proceedings of International Conference on Spoken Language Processing*, vol. 3, pp. 1099-1102, Sydney, Australia, 1998.

[17] J. Saunders: "Real Time Discrimination of Broadcast Speech/Music," *International Conference on Acoustics, Speech, and Signal Processing*, pp. 993-996, 1996.

[18] R. E. Schapire: "A Brief Introduction to Boosting," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[19] H. Schwenk: "Using Boosting to Improve a Hybrid HMM/Neural Network Speech Recognizer," *International Conference on Acoustics, Speech, and Signal Processing*, pp. 1009-1012, 1999.

[20] B. Ziół, S. Manandhar, and R. C. Wilson: "Fuzzy Recall and Precision for Speech Segmentation Evaluation," *Proceedings of 3rd Language & Technology Conference,* Poznan, Poland, 2007.