Content Based Retrieval of Music Videos

Sumandeep Banerjee

German Research Center for Artificial Intelligence
Erwin-Schroedinger-Strasse 57, 67663 Kaiserslautern, Germany
sumandeep.banerjee@gmail.com

Abstract. In Music Videos most of the relevant information exists in the audio track. We present a scheme of music video retrieval involving the feature extraction on the audio track, clustering the feature vectors, and an algorithm of querying, which returns the closest matching segment of a song, rather than the whole song itself.
Keywords - Music information retrieval, matching.

1 Introduction

Growing digital media collections have created the demand for more improved retrieval mechanisms. Significant amount of work has been done on music information retrieval as well as on video retrieval. But the problem of music video retrieval, where the relevant information is in the audio track, rather than the video has not yet been properly explored.

In this work, we present a scheme of retrieving songs from an entire collection of music videos. The query matching algorithm is such that, it is able to return the closest matching segment within the song, without significant computations, or searching using similarity measure throughout the entire length of the song.

In Section 2 we specify the feature extraction methodology employed. In Section 3 we discuss clustering of the data. The query matching algorithm is presented in Section 4. Section 5 presents an analysis of the computational complexity of the query algorithm. Section 6 concludes the paper.

1.1 Previous Work

Most of the work in Music Information Retrieval follow a common structure of feature extraction, followed by classification / indexing and finally a similarity measure with varying set of parameters. A detailed analysis of such methods, have been presented in [1]. Some of the methods are inclined towards classification [2], [3], while others, [6] concentrate on playlist generation based on similarity measures.

For feature extraction in most of the methods, as is in the present work, MFCCs are used. [2] proposes other additional parameters such as Spectral Centroid, Spectral Rolloff, Spectral Flux and Time Domain Zero Crossings. But these seem to have more usage in classification, rather that retrieval, and were not considered in the present work.

The statistical modeling of the MFCC distribution [1] is computed using a variety of ways. Some use K-means [6], [4], while others have employed Gaussian Mixture Models (GMMs) [4], [8], [7].

Finally, the distance measure is done using any of Euclidean, Earth Mover's Distance [6], KL divergence [3], Mahalanobis distance [3].

Almost all of the various works mentioned above are either interested in classifying the songs into music genres [2], [3], or on song recomendation [6], but none does mention ways of determining matching segments of song, rather than the whole song itself. This problem is addressed in the present work.

2 Feature Extraction

In a music video almost all relevant information is contained in the audio track. Hence, the feature extraction is performed on the audio track only. The audio track extraction is done using Media Coder [13]. Mel Frequency Cepstral Coefficients [9] have long been used as feature vectors in speech recognition. Logan [5] showed that the assumptions in computing MFCCs are valid for music modeling. Other works [6], [2], [3], have successfully employed MFCCs for music information retrieval.

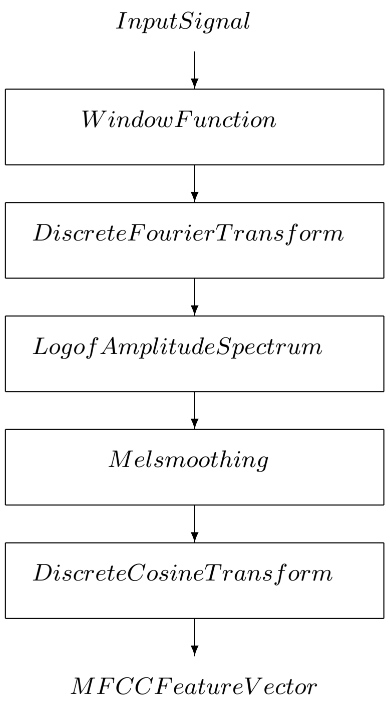
MFCC feature extraction is motivated by perceptual and computational considerations [5].The first step is to take short snippets of the audio signal, by applying a windowing function at fixed intervals. Each frame thus generated is a segment of the signal which is statistically stationary. The window function removes edge effects.

In the next step Discrete Fourier Transform (DFT) is applied on each frame. The logarithm of the amplitude signal is retained. Perceptual studies have shown that the amplitude is more important than the phase, and the perceived loudness of a signal is approximately logarithmic.The next step is to smooth the spectrum, and emphasize perceptually meaningful frequency components. The lower frequencies were found to be more perceptually important then the higher frequencies. This step is known as mel smoothing. The mel spectral vectors are highly correlated, thus to decorrelate the components, Discrete Cosine Transform (DCT) is applied. The transform domain output is our feature vector.

The input signal is split into frames of 11ms each, with an overlap of 6ms. The feature vector is represented by 13 MFCC coefficients, thereby generating feature vectors of 13 dimensions representing 11ms of signal. Such high resolution in time domain is generally not required. The MFCC feature vectors are further down-sampled to represent 250ms of data. This greatly reduces the number of feature vectors, without significant compromise in the retention of information about the signal.

3 Clustering

Querying by similarity measure over all the feature vectors in the database would be extremely computationally expensive. Thus, clustering is performed on the



$InputSignal$	$44.1KHz$
	$16bitMono$
$WindowFunction$	$Hamming$
$WindowSize$	$11ms$
$HopSize$	$6ms$
$FFTSize$	512
$MFCCFeatures$	13

Fig. 1. MFCC Feature Vector (a) Computational Steps (b) Parameters.

entire feature vectors using K-Means clustering. The number of clusters is chosen as $K = \sqrt{N}$, where N is the number of feature vectors in the song database [11], [12]. Alternatively, the number of clusters can also be chosen by the method presented by Pelleg and Moore in [10], but have been not used in the implementation, as this is beyond the scope of the present work. All the feature vectors are clustered and cluster centers are stored to be used for querying, thereby reducing the search complexity for a query from $O(N)$ to $O(K)$. This greatly improves the search efficiency.

4 Query Matching Algorithm

The query matching algorithm proposed is quite unique. The various steps involved in the analysis of the query and generation of matching segments of song, is explained one by one in this section.

4.1 Feature Extraction

The query audio signal is split into frames as explained above in the section:feature extraction. Each frame represents 250ms of temporal signal, and is denoted by a 13 dimensional MFCC feature vector.

```
Input : Query i.e., audio signal in time domain
Step 1: Compute MFCCsfor the query
Step 2: Do for each MFCC frame
a. Find nearest cluster center
b. Cluster members are potential hits
c. Collect all cluster members
Step 3: Group the hits (from step 2) by their song ID
Step 4: Do for each song
a. Identify contiguous / semi-contiguous segments of the song
b. Apply a smoothing function to the segments
Step 5: Rank the identified segments using the Hamming distance between
the original and the smoothed version of the segments
Output: Ranked list of song segments matching the query
```

Fig. 2. QueryDB - Query Matching Algorithm

4.2 Winner List

Each MFCC feature vector generated from the query is compared with all the cluster centers, and the nearest cluster center is determined using L_2 Norm (Euclidean). We get one winner cluster center per frame of the query audio signal.

The collection of cluster members (MFCC feature vector of frame) of the entire list of winner cluster centers contains all the frames having a possible match in the query snippet.

4.3 Matching Segment

The collection of frames in the winner list are grouped by the songs they belong to. The frames of a song which appear in the list are denoted by '1' and frames which do not appear are denoted by '0'. The sequence of 1's and 0's are sorted according to their temporal occurrence in the song. This sequence is called frame sequence of the song.

The next step is to identify contiguous / semi-contiguous sequences of 1's appearing in the frame sequence of the song, which may be deduced as a possible matching segment for the query.

4.4 Ranking of Matching Segments

Each contiguous sequence of 1's in the frame sequence is regarded as a matching song segment for the query snippet. In some sequences, there may be a few outliers of 0's in between long sequences of 1's. These occur when in some frame sequences some matches are missed due to noise etc. Also there could be solitary 1's giving false impression of a segment. These outliers break the contiguous nature of the frame sequence. To improve the performance of the segment matching, it is required that these outliers must be removed.

The smoothing of the frame sequence, to get rid of outliers, is done by applying autoregressive low-pass filter (ARLPF). The ARLPF masks small number of outlying 1's and 0's in the sequence, to produce better contiguous matching segments within each song. We have used a triangular ARLPF of length 11.

The rank of a matching segment is determined by measuring the degree of smoothness i.e. by the extent of it's contiguous nature. The matching segment before being smoothed by ARLPF is compared with the smoothed segment. The hamming distance between the two gives a numerical idea of it's smoothness. The hamming distance per frame, gives a normalized rank value of the smoothness of the matching segment. The lower the rank value, the segment is considered better matching.

The process of identification and ranking of matching segments is performed for each song identified in the hit list. The result is displayed in the increasing order of the rank values for each matching segment. The result specifies, the start and end times of the matching segment, and the song in which it appears.

5 Complexity Analysis

I this section we will present an analysis of the computational complexity of the proposed query matching algorithm. First we will begin with the definition of the parameters involved.
N : number of frames / feature vectors in the database
K : number of clusters
D : dimention of feature vector
n : number of frames in the query snippet

Some properties of the parameters worth noting are, $n << N$, $K = \sqrt{N}$, and $D = 13 - 40$.

The complexity of computing the list of winner cluster centers is $O(nKD)$. As $K = \sqrt{N}$, the algorithm is sub-linear with respect to the size of the database (N). In the worst case the number of winner cluster centers is n i.e. each frame of the query belongs to a different cluster. The winner cluster center list is sorted for easier further processing, using Quick Sort, which takes $O(n\log n)$ time.

The generation of the winner list requires a single iteration over the entire collection of cluster members of the winner clusters. The average number of members in each cluster is $\frac{N}{K}$. Let the number of winner clusters be $w = \frac{nN}{K}$. Thus, hit list takes $O(w)$ amount of time to be prepared.

Let's assume that the number of frames in a song identified in the winner list is p, where $p > n$ and $p << N$. The frame sequence can be prepared in time $O(p)$. The smoothing of the frame sequence using ARLPF takes $O(pg)$, where g is the order of the ARLPF.

6 Results and Conclusion

For testing purposes, a collection of 65 most popular video songs of 2005 were used. The total number of MFCC feature vector frames (N) for the database was 103,064. Each feature vector represented 250ms of audio signal. The number of clusters were taken to be $\sqrt{N}$ as 321.

Rank	Normalized Hamming Distance	Matching segment length (Sec)	Song ID	Start position (Sec)	End position (Sec)
1	6	42	37 Heartb	14	56
2	16	71	07 Smile.	37	108
3	16	34	07 Smile.	199	234
4	19	24	18 Leaf H	6	30
5	19	17	21 Us.wav	82	99
6	19	27	36 Breath	193	220
7	19	17	63 New Sl	406	423
8	21	42	25 Your E	41	83
9	21	27	36 Breath	162	190
10	21	28	49 Going	253	282
11	23	19	10b Hitch	270	289
12	23	17	53 Mushab	257	274
13	24	36	07 Smile.	162	198
14	26	18	32 Walk A	233	252
15	31	21	37 Heartb	130	151

Table 1. Query result for a typical query of 15 second snippet from song 37. Query time was 0.968 seconds

A typical query result is given in Table 1. For most of the test runs, the top ranked result, came from the song from which the query snippet was extracted. The segment of the result determined by the algorithm, contained the snippet along with the portion of the song most relevant to the timbral texture of the query.

We can see from the discussion in the previous section that complexity of all the steps in the algorithm are sub-linear with respect to the size of the database (N), and most are linear with respect to the query size (n). So, as our database is increased, there won't be significant increase in the querying time. This is quite good considering the fact, that we are not only choosing the best matching song, but also able to predict the position of occurrence of the query within the song.

There is lot of scope for further improvement, in areas like determining the number of clusters (K) [10], which greatly affects the complexity of almost all the steps. The ARLPF is central to the correctness / relevance of the result displayed. Varying the order and shape of the ARLPF could have significant effect on the results. Further research could be directed to explore various possibilities in these areas.

References

1. J.J.Aucouturier and F.Pachet: Improving timbre similarity: How high is the sky? Journal of Negative Results in Speech and Audio Sciences, 1(1), (2004).
2. G.Tzanetakis and P.Cook: Musical genre classification of audio signal. IEEE Transactions on Speech and Audio Processing, 10(5), (July 2002).
3. M.Mandel and D.Ellis: Song-level features and support vector machines for music classification. Proceedings of the Sixth International Conference on Music Information Retrieval, (2005).
4. A.Berenzweig, B.Logan, D.P.W.Ellis, and B.Whitman: Proceedings of the Fourth International Conference on Music Information Retrieval, (2003).
5. B.Logan: Mel frequency cepstral coefficients for music modelling. Proceedings of the First International Symposium on Music Information Retrieval, (2000).
6. B.Logan and A.Salomon: A music similarity function based on signal analysis. Proceedings of the IEEE International Conference on Multimedia and Expo, (2001).
7. V. Kulesh, I. Sethi, and P.V: Indexing and retrieval of music via gaussian mixture models. Proceedings of the 3^{rd} International Workshop on Content-Based Multimedia Indexing, (2003).
8. J.J.Aucouturier and F.Pachet: Music similarity measures: What's the use? Proceedings of the 3^{rd} International Conference on Music Information Retrieval, (2002).
9. L.R.Rabiner and B.H.Juang: Fundamentals of Speech Recognition. Prentice-Hall, (1993).
10. D.Pelleg and A.Moore: X-means: Extending k-means with efficient estimation of the number of clusters. Proceedings of the Seventeenth International Conference on Machine Learning, (2000).
11. U.Maulik and S.Bandyopadhyay: Performance Evaluation of Some Clustering Algorithms and Validity Indices. IEEE Transactions on PAMI, 24(12):16501654, (2002).
12. T.Kaster, V.Wendt and G.Sagerer: Comparing Clustering Methods for Database Categorization in Image Retrieval. Proceedings, Lecture Notes in Computer Science, Springer-Verlag (2781):228235, (2003).
13. Media Coder: http://www.rarewares.org/mediacoder/