Agglomerative Hierarchical Clustering For Musical Database Visualization and Browsing

Aristomenis S. Lampropoulos[1] and George A. Tsihrintzis[2]

Department of Informatics
University of Piraeus
Piraeus 185 34
GREECE

[1]Aristomenis@rainbow.cs.unipi.gr
[2]Geoatsi@unipi.gr

Abstract. We propose an automated system which is based on agglomerative hierarchical clustering and automatically organizes a collection of music files according to musical surface characteristics (e.g., zero crossings, short-time energy, etc.) and tempo. The system architecture is based on three distinct yet inter-related processes, namely: (a) Preprocessing (format normalization and extraction of fifteen features per song), (b) Retrieval and (c) Browsing. The organization of music files is visualized via a dendrogram in which every leaf of the tree represents a music file and musical similarity decreases with height. Additionally, our system is complemented by a Query-By-Example (Q.B.E.) retrieval subsystem, the results of which are also returned to the user in dendrogram form.

Keywords: Content-based retrieval, musical database, query-by-example

1 Introduction

With the huge increase in the availability of digital music, it is becoming increasingly important to automate the task of querying and browsing a database of musical pieces. At present, music data are usually organized on the basis of textual meta-information. The best-known example of this organization is probably the ID3 format, an extension to the popular MP3 format which allows the user to add tags to the beginning or end of the file, containing such information as song title, artist, album name, genre, etc. Despite its extended capabilities, ID3-based systems still suffer from serious drawbacks. For example, the textual description of audio content is subjective and the relevant meta-data have to be entered and updated manually, which implies significant effort in both creating and maintaining the music database. There-

fore, it is expected that extracting the information from the actual music data through an automated process could overcome some of these problems [3].

The goal of our project is to create a simple and friendly method for the user to organize his/her music files automatically. Specifically, we propose such an automated system which is based on agglomerative hierarchical clustering and automatically organizes a collection of music files according to musical surface characteristics (e.g., zero crossings, short-time energy, etc.) and tempo. The system architecture is based on three distinct yet inter-related processes, namely: (a) Preprocessing (format normalization and feature extraction), (b) Retrieval and (c) Browsing. The organization of music files is visualized via a dendrogram in which every leaf of the tree represents a music file and musical similarity decreases with height. Additionally, our system is complemented by a Query-By-Example (Q.B.E.) retrieval subsystem, the results of which are also returned to the user in dendrogram form.

The paper is organized as follows: Section 2 describes a music/sound feature extraction process, which maps each song to a feature vector of low dimensionality (fifteen features per song). An overview of the agglomerative hierarchical clustering is given in Section 3 and Section 4 describes the proposed system architecture. For illustration purposes, implementation information and an illustrative demonstration are given in Section 5, while conclusions and some directions for future research are given in Section 6.

2 Music/Sound Classification Characteristics

To be able to search through a collection of music pieces and make observations about the similarity between objects that are not directly comparable, we must transform raw data at a certain level of information granularity. *Information granules* refer to a collection of data that contain only essential information. Such granulation allows more efficient processing for extracting features and computing numerical representations that characterize a music file. As a result, the large amount of detailed information of each song is reduced to a limited collection of features. Each feature is a vector of low dimensionality, which captures some aspects of a song and can be used to determine song similarity.

An extensive variety of features can be extracted from raw audio data using either their time domain or the frequency domain (spectral) representation [4], [5], [6], [7], [8], [13]. In our system, we used a 15-dimensional feature vector, comprised of the following features: Mean Centroid, Mean Roll-Off, Mean Zero-Crossings, Energy, Spectral Flux (5 spectral bands), Short Time Fourier Transform (5 spectral bands) and Tempo. The mean centroid, roll-off and zero-crossings features are calculated over a "texture window" of duration of 5 seconds consisting of 100 "analysis windows" of duration of 50 milliseconds. On the other hand, the short time Fourier transform (STFT) feature calculation is based on a Fast Fourier Transform (FFT) algorithm [1], [3]. In the following, we describe the feature computation in detail.

2.1 Musical Surface Features

For pattern recognition purposes, we can use the statistics of the spectral distribution over time and represent the "musical surface" [5], [6], [7], [8], [13]. Some of these statistics are defined next.

Spectral Centroid. This feature reflects the brightness of the audio signal. It is computed as the balancing point (centroid) of the spectrum. It can be calculated as

$$C = \frac{\sum_{n=1}^{N} M_t[n] \cdot n}{\sum_{n=1}^{N} M_t[n]} , \quad (1)$$

where $M_t[n]$ is the magnitude of the Fourier Transform at Frame *t* and frequency bin *n*.

Spectral Rolloff. This feature describes the spectral shape. It is defined as the frequency *R* corresponding to *r*% of the magnitude distribution. It can be seen as a generalization of the spectral centroid, as the spectral centroid is the roll-off for r = 50%. In our system, we used a roll-off value of *r* = 95%.

$$\sum_{n=1}^{R} M_t[n] = r \sum_{n=1}^{N} M_t[n] . \quad (2)$$

Spectral Flux. This feature describes how the frequency evolves with time. It is computed as the difference of the magnitude of the short-time Fourier transform between the current and the previous frame. In other words, it is a measure of local spectral change and is defined as

$$SF = \sum_{n=1}^{N} \left(N_t[n] - N_{t-1}[n] \right)^2 , \quad (3)$$

where $N_t[n]$ is the normalized magnitude of short-time Fourier transform at window *t*.

Zero-Crossings. A zero-crossing occurs when successive samples in a digital signal have different signs. The corresponding feature is defined as the number of time domain zero-crossings of the signal. This feature is useful to detect the amount of noise in a signal. It can be calculated as

$$Z_n = \sum_m \left|\text{sgn}[x(m)] - \text{sgn}[x(m-1)]\right| w(n-m) \ , \tag{4}$$

where

$$\text{sgn}[x(n)] = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases} , \tag{5}$$

and

$$w(n) = \begin{cases} \frac{1}{2}, & 0 \leq n \leq N-1 \\ 0, & otherwise. \end{cases} . \tag{6}$$

Short–Time Energy Function. The short-time energy of an audio signal is defined as

$$E_n = \frac{1}{N} \sum_m \left[x(m) w(n-m)\right]^2 \ , \tag{7}$$

where

$$w(n) = \begin{cases} 1, & 0 \leq n \leq N-1 \\ 0, & otherwise. \end{cases} . \tag{8}$$

In Eqs.(7) and (8), *x(m)* is the discrete-time audio signal, *n* is the time index of the short time energy and *w(n)* is a rectangular window. This feature provides a convenient representation of the temporal evolution of the amplitude variation.

2.2 Rhythm Feature

Tempo. In order to extract tempo we use the beat detection algorithm provided at the site [14]. This algorithm is quite similar to a beat detection algorithm developed by the MIT Media Lab. It amounts to emphasizing the sudden impulses of sound in the song and then finding the fundamental period at which these impulses appear. This is accomplished by breaking the signal into frequency bands, extracting the envelope of the resulting frequency–banded signals, differentiating them to emphasize sudden changes in sound, and running the signals through a bank of comb filters and choosing the highest energy result as tempo. A drawback of this algorithm is that it does not account for the possibility of a varying tempo in a music piece. Specifically, the algorithm extracts a piece of music of duration of 2.2 seconds from the middle section of a song, analyzes its tempo, and assumes that the computed tempo is associated with the entire music piece.

3 Agglomerative Hierarchical Clustering

Hierarchical clustering is widely used to detect patterns in multidimensional datasets. Hierarchical cluster analysis is a statistical method for finding relatively homogeneous clusters (groups of data) on the basis of measured characteristics. It starts with each piece of data put in a separate cluster and then iteratively combines clusters into wider ones, thus reducing their number. The process is terminated when a desired number of clusters is reached. This hierarchical clustering process can be visualized in a dendrogram form, where each step in the clustering process is illustrated by a join in the tree.

Thus, the agglomerative hierarchical clustering algorithm [2] is summarized as follows. Let us assume that we want to cluster n data points. Then, $n(n\text{-}1)/2$ similarity values correspond to every possible pair of the n data points.

1. Initially, each data point occupies a cluster of its own and, thus, there are n clusters at first.

$$c = n, \quad X_i = \{x_i\}, \ i = 1,2,...,n \ . \tag{9}$$

2. If k is the desired minimum number of clusters and

$$c \le k, \tag{10}$$

then stop, else find a pair of clusters X_i and X_j whose similarity value is the highest.

3. Merge the pair into a single cluster and decrease the cluster number c by one, so that c=c-1. Compute distances between the new cluster and each of the existing clusters.
4. Go to step 2, until all items are clustered into a single cluster of size n.

There are many possible choices in updating the similarity values in step 4. Among them, most common ones are (1) single-link, (2) average-link, and (3) complete-link. In single link clustering (also called connectedness or minimum method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.

$$d_{\min}(X_i, X_j) = \min \left\| \underline{x} - \underline{x'} \right\| \text{, where } \underline{x} \in X_i \text{ and } \underline{x'} \in X_j \ . \tag{11}$$

3.1 Distance Metrics

We use two distance measures [4] in our system, which are common in agglomerative hierarchical clustering.

Euclidean Distance. The vector of features for a particular song is treated as point in an *n*-dimensional Euclidean space. The model assumes that two songs, *P* and *Q*, sound similar if their feature sets *p* and *q*, respectively, are within distance ε, as computed by the Euclidean distance

$$D_E^2(p,q) = \sum_{i=1}^{N} [p(i) - q(i)]^2 \quad , \tag{12}$$

where $p(i)$ denotes the i^{th} component of *p* and *i* assumes the integer values 1, …, |*p*|.

Cosine Distance. The user is given the possibility to use, instead of the Euclidean distance, the cosine distance as an alternative similarity metric.

$$D_C = \frac{\sum_{i=1}^{N} p(i)q(i)}{\sqrt{\sum_{i=1}^{N} p(i)^2 \times \sum_{i=1}^{N} q(i)^2}} \; . \tag{13}$$

This measure computes the cosine of the angle between feature sets *p* and *q* of two songs *P* and *Q*.

4 Proposed System

In this section we describe the system architecture which is based on three distinct yet inter-related processes, namely: (a) Preprocessing (format normalization and feature extraction), (b) Retrieval and (c) Browsing.

4.1 Preprocessing

We collected a set of 110 files containing Greek music. Each music piece is allowed to be stored in any audio file format, such as *.mp3*, *.au*, or *.wav*. In order to extract features, first we apply format normalization. We decode each music file to raw Pulse Code Modulation (PCM), converting it to the .wav format with audio CD quality and resolution of 16 bit samples at a sampling rate of 44.1 KHz. Next, we reduce stereo sound quality to mono quality. Reducing stereo to mono implies no significant information loss as the two channels have the same samples.

From each file, a randomly chosen sample of duration of 5 seconds is used as input to the feature extraction module. The actual features computed by our system are the running means over a number of analysis windows of the features described in Section 2. In short time audio analysis, the signal is broken into small, possibly overlap-

ping temporal segments of duration of 50 milliseconds and each segment is processed separately. These segments are called ‘‘analysis windows’’ and need be short enough for the frequency characteristics of the magnitude spectrum to be relatively stable. The term ‘‘texture window’’ describes the longest window that is necessary to identify music texture. This corresponds to the minimum amount of sound needed to identify music texture and was equal to 5 seconds in our system.

4.2 Query-By-Example Music Similarity Retrieval

Query-by-example (QBE) music similarity retrieval is implemented on the basis of the musical content features using the single feature vector representation for each file. As the computed features represent various characteristics of the music, feature vectors that are similar (that is, close together in feature space) are expected to be similar in perception as well. Therefore, the QBE approach searches for data that are similar to the given query. An example of QBE query is “Which records in our audio database are similar to a given audio query clip”.

QBE can be categorized into two subtypes according to the application approach: **Search By Restricted Format (SBRF)** and **Search By Unrestricted Format (SBUF)**. A search by restricted format query looks for data that are *globally* similar to the query, without paying attention to the scale of query. In other words, all records in a database must be resampled into a uniform sample rate. An example of a SBRF query is “Which are the data items that are similar to the given query, as a whole”. In contrast, a SBUF query searches for data with local similarities to the query. An example of a SBUF query is “List all items that contain ‘‘portions’’ which are similar to the given query.” In our system we have implemented the query by example in the search by restricted format, using as a distance metric either the Euclidean distance or the cosine distance.

4.3 Browsing and Visualization of the Musical Database

Visualization techniques are very common in many scientific domains. They take advantage of the powerful pattern recognition abilities of the human visual system to reveal similarities and patterns in data. Visualization is more suitable to disciplines that are exploratory in nature where there are large amounts of data to be analyzed.

Routinely, the information access problem presumes a query, which summarizes the user’s expression of an information need. The system task is, then, to search a collection for music files that match this need. However, the user may not be looking for something specific at all, but rather may wish to discover the general information content of the music collection. In fact, access to a musical database covers an entire spectrum: on one hand there is the need for a narrowly specified search for a particular music file, while on the other hand there is a need for a browsing session with no well-defined goal, but to satisfy the curiosity to learn more about the database.

Therefore, an interface for visualization and interaction with large audio collections is particularly useful tool. A browser ought to be able to visualize music files in an efficient way, offering first a database overview and then zooming on detail and

filtering on demand. Thus, the dendrogram form of a browser provides a good solution for musical database browsing and visualization. The dendrogram browser the audio files of a collection in a tree representation. Tree browsers, such as the window explorer, are familiar to users for file organization and navigation.

Moreover, the cluster tree allows users to view information at any level of granularity as the dendrogram illustrates the entire process of agglomerative hierarchical clustering. The dendrogram takes advantages of the superb visual capabilities of human users and enables them to spot interesting patterns, relations and dominant clusters. The tree structure allows the user to navigate efficient as he/she can drill down or roll up the cluster tree, at any particular level of details.

Each audio file is mapped to a tree leaf, while each tree node represents an entire cluster and each link represents a parent-child relationship. The links between objects are represented as upside down U-shaped lines. The height of each U-shaped line indicates the similarity distance between objects on the Y-axis.

In order to determine the natural cluster division in our dataset, we compare the height of each link in a cluster tree with the heights of neighboring links at lower levels in the tree. On the one hand, if a link is approximately at the same height with neighboring links, this indicates that there are similarities between objects joined at this level of hierarchy. If the height of a link differs from neighboring links, this indicates that there are dissimilarities between the objects at this level of the cluster tree and the link is said to be inconsistent with the links around it. Highly similar nodes or subtrees have joining points that are farther from the root.

Consequently, browsing can be used as a complementary search mechanism, since the dendrogram allows the user to view the music files that belong to the same cluster and have a high degree of similarity.

5 Implementation

The system has been developed using *MatLab* as the computing engine for implementing the feature extractor and building a dynamically-linked library (dll) with the *Com-Builder* tool. On the other hand, the user interface was developed in the *.NET* framework using the *C#* and *VB .NET* languages. Extracted features and other metadata of music files are stored on *SQL SERVER 2000 RDBMS*. The entire architecture of the system is illustrated in Figure 1, while the appearance of the interface for QBE and browsing is shown in Figures 2 and 3, respectively.

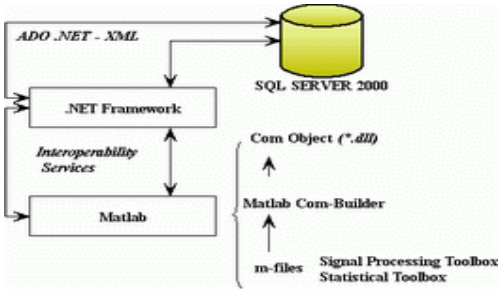

Fig. 1. *System architecture*

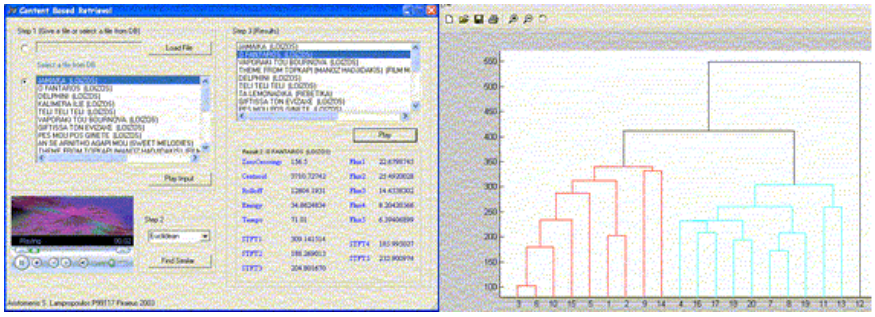

Fig. 2. *Query By example interface.* The user is allowed to select a music file and the distance measure. The system produces a playing list with twenty (20) most similar files that are contained in the database. The results are presented in dendrogram form

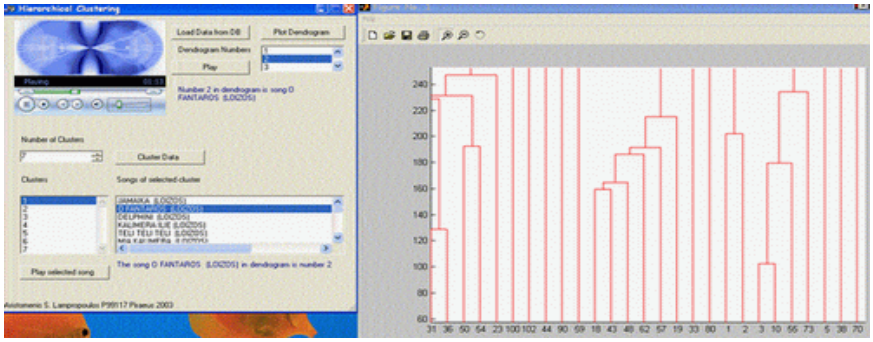

Fig. 3. *Browser.* The user selects a music file and then the system focuses on the specific subtree of the dendrogram

6 Conclusions – Future work

In this paper, we presented a content-based music retrieval system which uses agglomerative hierarchical clustering as an efficient method for visualization and browsing of a musical database and automatically organizes a collection of music files according to musical surface characteristics and tempo. We plan to improve further the clustering accuracy of our system by considering more music features such as rhythmic characteristics, adding new hierarchical grid clustering algorithms, applying new multidimensional visualization techniques and examining their efficiency. Further extensions of our system may follow the avenues of semantic content-based retrieval and related system architectures. Another avenue of future work may be in the direction of application of the proposed systems in the development of intelligent tutoring systems that may model the knowledge of Greek music. This and other work is currently in progress and will be presented at a later occasion.

References

1. Tsihrintzis, G. A., Douligeris, C.: Principles and Applications of Signals and Systems. Varmar Publications (2003)
2. Tsihrintzis, G. A.: Pattern Recognition. University of Piraeus (2002)
3. Steinmetz, R., Nahrstedt, K.: Multimedia Fundamentals Volume 1, Media Coding and Content Processing. Prentice Hall (2002)
4. Foote, J.: Content based retrieval of music and audio. Multimedia Storage and Archiving Systems II, In Proceedings of SPIE Volume 3229, pp. 138-147 (1999)
5. Foote, J.: An overview of audio information retrieval. Multimedia Systems, 7(1):2-10 (1999)
6. Kosina, K.: Music Genre Recognition. PhD thesis, Hagenberg (2002)
7. Tzanetakis, G.: Manipulation, Analysis and Retrieval Systems for Audio Signals. PhD thesis, Princeton University (2002)
8. Tzanetakis, G.: Automatic Musical genre classification of audio signals. In Proceedings of ISMIR2001 (2001)
9. Zhang, Tong, Jay Kuo, C.C.: Hierarchical system for content-based audio classification and retrieval. In Proceedings of International Conference on Acoustic, Speech, Signal processing. Volume 6, pp. 3001-3004 (1999)
10. Yang, C.: Music database Retrieval based on spectral similarity. In Proceedings of ISMIR2001 (2001)
11. Liu, Z., Huang, Q.: Content-based indexing and retrieval by example in audio. ICME (2000)
12. Welsh, M., Borisov, N., von Behren, R., Woo, A.: Querying large collections of music for similarity. Technical report UCB/CSD00-1096, U.C. Berkeley, Computer Science (1999)
13. Deshpade, H., Unjung Nam, Singh, R.: MUGEC Automatic music genre classification. Technical report, Standford University (2001)
14. Cheng, K., Nazer, B., Uppulury, J., Verret, R.: Beat detection algorithm. http://www.owlnet.rice.edu/~elec30/beatalgo.html