

Automatically Analyzing and Organizing Music Archives

Andreas Rauber, Markus Frühwirth

Department of Software Technology, Vienna University of Technology
Favoritenstr. 9 - 11 / 188, A-1040 Wien, Austria
<http://www.ifs.tuwien.ac.at>

Abstract. We are experiencing a tremendous increase in the amount of music being made available in digital form. With the creation of large multimedia collections, however, we need to devise ways to make those collections accessible to the users. While music repositories exist today, they mostly limit access to their content to query-based retrieval of their items based on textual meta-information, with some advanced systems supporting acoustic queries. What we would like to have additionally, is a way to facilitate exploration of musical libraries. We thus need to automatically organize music according to its sound characteristics in such a way that we find similar pieces of music grouped together, allowing us to find a classical section, or a hard-rock section etc. in a music repository.

In this paper we present an approach to obtain such an organization of music data based on an extension to our *SOMLib* digital library system for text documents. Particularly, we employ the *Self-Organizing Map* to create a map of a musical archive, where pieces of music with similar sound characteristics are organized next to each other on the two-dimensional map display. Locating a piece of music on the map then leaves you with related music next to it, allowing intuitive exploration of a music archive.

Keywords: Multimedia, Music Library, Self-Organizing Map (SOM), Exploration of Information Spaces, User Interface, MP3

1 Introduction

Currently, we are experiencing a tremendous increase in the amount of music being made available in digital form. Following the sweeping success of music swapping systems such as Napster, a further rise in the amount of music distributed via electronic archives is to be expected due to the current move of music industry towards electronic distribution of their products. As a consequence we may expect to find even larger archives containing innumerable pieces of music from a variety of genres. With the creation of large audio collections, however, we need to devise ways to make those collections accessible to the users. When talking about multimedia content we have to differentiate between the different forms in which content is made available, i.e. textual form, audio, video, animations, etc. While automatic handling and analysis of textual data has a long history in (textual) information retrieval, the more prominent types of multimedia information still rely heavily on manual

indexing and description to facilitate automatic content-based access later on. We thus find these kinds of metadata descriptors to form one of the core entities in the MPEG-7 standard [16]. However, due to the diversity and complexity of multimedia data we will, for the work described in this paper, limit our discussion to audio data.

When talking about interfaces to electronic music archives we may basically distinguish between four primary modes of access, which can be summarized as (1) database-oriented access allowing search via metadata; (2) text-based access searching the text of songs for particular phrases; (3) searching for a particular melody or tune; or (4) looking for a particular type of music, i.e. for titles within a specific musical genre.

Most music repositories existing today limit access to their content to the first approach, allowing query-based retrieval of their documents based on textual information, i.e. mainly via metadata about the documents, such as title, composer, artist, or band information, the label, and so on. This allows users to conveniently locate a particular piece of music when the necessary meta-information is available, satisfying a large percentage of users requests.

Slightly more challenging is the task of finding a particular piece of music based on the lyrics. This means of access requires the transcripts of texts to be available in electronic form, which has to be created mostly manually, with only limited support from speech recognition programs possible in the given setting.

The third way of searching music archives is based on melodies, which form a very natural approach to such collections. In this case, the input to the system is formed by a melody taken from a recording or hummed by a user, with the system trying to extract melody scores and matching those with the tunes in the collection.

Yet, all of these approaches address searches for a particular title, i.e. situations where a users knows more or less exactly what he or she is looking for. None of the three approaches naturally supports browsing of a collection, searching for a certain type of music, rather than for a specific title. Still, this approach to exploring music archives is one of the most prominent in conventional record stores, where customers find titles organized primarily by genre, with only subsequent alphabetical sorting by artists within the various sections. This allows users to browser through their sections of interest to find new pieces of music, reflecting an exploratory approach of music search as opposed to a direct search for a particular title. A similar form of organization is found with most people's private record collections, which commonly are organized by types of music as well, making it easier to pick the kind of music one would like to hear in a particular situation.

The need to support this kind of interaction resulted in the creation of manually tendered groupings of music files in most electronic music repositories, where artists file their music according to certain genres, such as Pop, Classic, Jazz, Punk, Blues, etc., mimicking the way music is organized in conventional music stores or libraries. Yet, when analyzing these genres we find them to be both rather subjective, and sometimes hard to interpret, as the variations within each genre are considerably large and sometimes not too well defined. The music archive *mp3.com* [14] for example, list its titles under 350 different genres, organized in up to four hierarchical layers. Since titles are commonly filed into such a hierarchy by different people, the

sound characteristics of titles of the same genre often vary a lot, making the classification a rather weak guideline to match to one's own musical taste. Furthermore, the manual classification task involved renders this approach difficult to support by many large archives with frequently changing collections.

What we would like to have, in this context, is a way to automatically organize music according to its sound characteristics in such a way, that we find similar pieces of music grouped together. This would allow us to find, e.g., a classical section, or a hard-rock section in a music repository, supporting an exploratory interface to audio collections. In this paper we present an approach to obtain such an organization of music data using a neural network approach which creates a map of the titles in a music archive. The task is comparable to content-based organization of textual documents, i.e. the grouping of texts according to the subject they deal with. We thus adapted the concepts of our *SOMLib* digital library system [19], a system providing content-based organization of text documents, to include audio data. We employ the *Self-Organizing Map (SOM)* [9] to create a map of a musical archive, where pieces of music sounding similar are organized next to each other on the two-dimensional map display. Locating a piece of music on the map then leaves you with similar music next to it, allowing intuitive exploration of a music archive.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work in the field of music retrieval. This is followed by a presentation of our approach to automatic organization of music archives by sound similarity in Section 3, covering feature extraction, the principles of the *Self-Organizing Map*, and the two-layered architecture used to organize music. Experimental results organizing an archive of MP3 music are presented in Section 4, followed by some conclusions as well as an outlook on future work in Section 5.

2 Related Work

Analysis of audio data has a long history, with the focus of work in this area ranging from spectra identification of certain instruments, via voice recognition, to fault detection in electric drives. With respect to music classification, we can distinguish between two main approaches, relying on frequency spectra analysis, or on melody analysis, respectively. Most melody-based approaches may be seen as an analogy to information-retrieval in text-based systems. Queries are either sung, or hummed, with the system extracting characteristic melody features and trying to match those with the audio files in the archive. In this setting, most works rely on the Musical Instruments Digital Interface (MIDI) file format [13], representing the scores for a series of instruments, rather than the sound itself. The scores have to be interpreted to synthesize the sound as such, making it a descriptive format for how to produce sound, rather than an audio file format itself. Yet, it has huge benefits in terms of melody retrieval, as it reduces the complexity of melody-based retrieval somewhat to the problem of matching symbolic musical notes. Hawley [8] developed a system that allows a note sequence being entered via a MIDI keyboard. It then searches the tunes whose beginnings exactly match the input.

To facilitate acoustic querying of MIDI data, the transcription of acoustic input into symbolic musical notes is highly challenging, as it requires the system to make up for all kinds of impreciseness, such as badly sung queries, rhythmic and melodic differences between various versions of the same melody, etc. Ghias [7] transforms a hummed query into a series of strings of tokens U , D , and S , representing up, down, and the same musical note as the previous one, using subsequent approximate string-matching for retrieval. One of the most prominent representatives of this kind of systems is the New Zealand Musical Digital Library [1, 12]. Apart from a variety of text-based searches within metadata records or the lyrics of songs, this system also allows the retrieval of tunes based on hummed queries. Users can specify their trust in their own capabilities of maintaining the right pitch, or rhythm, with the system then trying to match the hummed melody with the tunes in the collection. A similar system is reported in [21].

Yet, all of these systems focus on the retrieval of music data based on searches. We thus find a separate stream of research addressing analysis and classification of musical data. These mostly concentrate on the extraction of characteristic features, such as loudness, pitch, dynamics, and other sound characteristics. These are compared with reference vectors in a database to provide corresponding classification, allowing, for example, the distinction between different kinds of sounds such as applause or laughter. As one of the seminal works in this field we find the works by Wold et al. [22], as well as a similar system described by Foote [6], which uses a tree-structured vector quantizer, identifying, e.g. different types of speakers. Pfeiffer describes a system capable of violence detection in video soundtracks [17]. Other systems working directly on acoustic audio data in this context concentrate on beat tracking, trying to identify regular rhythmic structures. Works in this direction are reported in [4, 15].

In the same spirit we find works addressing instrument detection based on frequency analysis. Special filters are being applied to extract and put more weight on frequencies characteristics for a specific instrument. Cusi et al. [2] describe a system for timbre classification to identify 12 instruments in both clean and degraded conditions. Similar to the works described in this paper, a *Self-Organizing Map* is used to cluster the resulting feature vectors. A similar approach, yet incorporating psycho-acoustic models for frequency transformation are described by Feiten [5], again using *SOMs* for clustering the data.

3 The SOMeJukebox

3.1 Basic principles

What we would like to have is a digital library system capable of automatically organizing musical data based on its sound characteristics. We have thus extended our *SOMLib* digital library system [19, 20] to incorporate audio data, resulting in the *SOMeJB*, i.e. the *SOM-extended Jukebox* system. The *SOMLib* system is capable of automatically organizing a collection of text documents according to their content. The *Self-Organizing Map (SOM)*, a popular unsupervised neural network model,

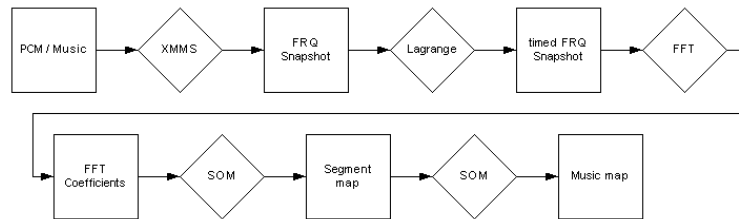


Fig. 1. SOMeJukeBox System: feature extraction, conversion, and 2-level SOM training is used to provide a topically sorted map of the various documents, using word histogram representations during an unsupervised training process.

By including a special feature extraction process we can apply the systems principles to the automatic organization of musical data. Figure 1 provides an overview of the resulting *SOMeJB* system. As the characteristics of a piece of music vary, a two-layered clustering procedure is used to first create a cluster of music segments of about 3 to 5 seconds length. In a second step, the clustering results of the various musical segments are used to analyze and cluster pieces of music according to their overall similarity.

In a nutshell, the principles of the system can be described as follows. We use *XMMS*, a popular open source media player, to extract frequency spectra from music data such as MP3 or Wave files. As these frequency spectra are not provided in evenly spaced time intervals, we use Lagrange transformation to obtain timed snapshots. This is followed by a Fast Fourier Transformation (FFT) across the segments for a selected set of frequency spectra to obtain Fourier coefficients modeling the dynamics. These feature vectors are used to train a *SOM* of music segments. Segments with similar sound characteristics are located next to each other on the map, while highly differently sounding segments are found in sections far apart on the map. Quite frequently we also find pieces of music to show two very distinct sound characteristics in their verses and chorus parts, thus having their segments spread across different parts of the map. In a second step, a unique feature vector is created for each piece of music based on the cluster distribution of its segments. These vectors are again fed into a *SOM* to create a map of the music. The individual steps are described in more detail in the following subsections.

3.2 Feature extraction and preprocessing

Music comes in a variety of file formats such as MP3, WAV, AU, etc., all of which basically store the sound information in the form of pulse code modulation (PCM) using a sampling rate of 44.1 kHz. The analog sound signal is thus represented by 44.100 16 bit integer numbers per second, which are interpreted by media players to reproduce the sound signal. In order to be able to compute similarity scores between musical tunes, a feature vector representation of the various pieces of music needs to be created, which can further be analyzed by the *SOM*.

Starting with any popular music file format, most media players, such as the open source X Multimedia System (*XMMS*) are capable of splitting this data stream

into several frequency bands. Using XMMS, the signal is split into 256 frequency bands, with approximately one sample value every 20 to 25 ms each. Since not all frequency bands are necessary for evaluating sound similarity, and in order to reduce the amount of data to be processed, a subset of 17 frequency bands (i.e. every 15th frequency band) is selected for further analysis, covering the whole spectrum available. In order to capture musical variations of a tune, the music stream is split into sections of 5 seconds length, which are further treated as the single musical entities to be analyzed. While basically all 5-second sequences could be used for further analysis, or even overlapping segments might be chosen, experimental results have shown that appropriate clustering results can be obtained by the *SOM* using only a subset of all available segments. Especially segments at the beginning as well as at the end of a specific piece of music can be eliminated to ignore fade-in and fade-out effects. Furthermore, our results show that choosing every second to third segment, i.e. a 5-second interval every 10 to 15 seconds, provides sufficient quality of data analysis.

The intervals between the frequency snapshots provided by the player varies with the system load and can thus not be guaranteed to occur at specified time intervals. We thus have a set of amplitude/timestamp values about every 20 to 25 ms in each of the 17 selected frequency bands. In order to obtain equi-distant data points, a Lagrange interpolation is performed on these values as provided in Expression 1, where $f(z_k)$ represents the amplitude of the sample point at time stamp z_k for $n + 1$ sample points.

$$P_n(z) = \sum_{k=0}^n \left(\prod_{l=0, l \neq k}^n \frac{z - z_l}{z_k - z_l} \right) f(z_k) \quad (1)$$

As a result of this transformation we now have equi-distant data samples in each frequency band. The resulting function can be approximated by a linear combination of sine and cosine waves with different frequencies. We can thus obtain a closed representation for each frequency band by performing a Fast Fourier Transformation (FFT), resulting in a set of 256 coefficients for the respective sine and cosine parts. Combining the 256 coefficients for the 17 frequency bands results in a 4352-dimensional vector representing a 5-second segment of music. These feature vectors are further used for training a *Self-Organizing Map*.

3.3 Self-organizing maps

The *Self-Organizing Map (SOM)* [9] is an unsupervised neural network providing a mapping from a high-dimensional input space to a usually two-dimensional output space while preserving topological relations as faithfully as possible. The *SOM* consists of a set of i units arranged in a two-dimensional grid, with a weight vector $m_i \in \mathfrak{R}^n$ attached to each unit. Elements from the high dimensional input space, referred to as input vectors $x \in \mathfrak{R}^n$, are presented to the *SOM*, and the activation of each unit for the presented input vector is calculated. Commonly, the Euclidean distance between the weight vector of the unit and the input vector serves as the

activation function. In the next step the weight vector of the unit showing the highest activation (i.e. the smallest Euclidean distance) is selected as the ‘winner’ and is modified as to more closely resemble the presented input vector. Pragmatically speaking, the weight vector of the winner is moved towards the presented input signal by a certain fraction of the Euclidean distance as indicated by a time-decreasing learning rate α . Thus, this unit’s activation will be even higher the next time the same input signal is presented. Furthermore, the weight vectors of units in the neighborhood of the winner as described by a time-decreasing neighborhood function ϵ are modified accordingly, yet to a less strong amount than the winner.

This learning procedure finally leads to a topologically ordered mapping of the presented input signals. The weight vectors of a trained map serve as prototype vectors, or cluster centroids. Similar input data is mapped onto neighboring regions on the map. While each feature vector is mapped onto its most similar unit, we may further use the mapping distance, i.e. the Euclidean distance between the feature vector and the unit’s weight vector, as an indicator of how well the feature vector corresponds to the characteristics of the cluster.

SOM based architectures found wide appreciation in the field of text clustering [10, 11] due to their capabilities of handling very high-dimensional feature spaces as well as being able to cope with the inherent noise in the data representation. These characteristics also make the *SOM* a particularly suitable tool for music data clustering [2, 5], where we find rather similar situations. The *GHSOM* [3], an extension to the basic *SOM* algorithm furthermore allows the detection of hierarchical clusters, making it a suitable interface to explore large data collections [18]. For the given experimental setting, however, we will rely on the standard *SOM*.

3.4 Music clustering: segments and pieces of music

The feature vectors representing music segments can be thought of as data points in a 4352-dimensional space, with similar pieces of music, i.e. segments exhibiting similar frequency spectra and thus similar FFT coefficients, being located close to each other. Using the *SOM* to cluster these feature vectors, we may expect similar music segments to be located close to each other in the resulting map display.

On the resulting segment *SOM* the various segments are scattered across the map according to their mutual similarity. This allows, for example, pieces of music touching on different musical genres to be located in two or more different clusters, whereas rather homogeneous pieces of music are usually located within one rather confined cluster on the map. While this already provides a very intuitive interface to a musical collection, a second clustering may be built on top of the segment map to obtain a grouping of pieces of music according to their overall characteristics.

To obtain such a clustering, we use the mapping positions of the segments of a piece of music. We create a feature vector representation for each piece of music using the location of its segments as descriptive attributes. Given an $x \times y$ *SOM* we create an $x \cdot y$ dimensional weight vector, where the attributes are the (coordinates of) the units of the segment *SOM*. Each vector attribute represents the number of segments of a particular piece of music mapped onto the respective unit in the *SOM*. Consider a piece of music *A* consisting of 7 segments, three of which are

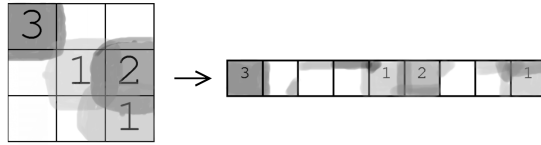


Fig. 2. Creating 2nd-level vectors based on segment distribution

mapped onto unit $(0/0)$ ¹ in the upper left corner of a 3×3 map, two segments on unit $(2/1)$, and one segment on the neighboring units $(1/1)$ and $(2/2)$, respectively, as depicted in Figure 2. The attributes of the resulting 9-dimensional feature vector of the song are basically set to the according values $(3\ 0\ 0\ 0\ 1\ 2\ 0\ 0\ 1)^T$. Subsequent norming to unit length makes up for length differences of songs.

Instead of directly using the number of segments mapped onto a specific unit as the attribute of the newly created input vector for a given piece of music, we may improve data representation by incorporating information about the similarity of a given segment with the weight vector of the unit it is mapped onto. As the weight vector serves as a cluster prototype, we can use the mapping distance between a segments feature vector and the unit’s weight vector to give higher weights to segments that are very similar to the cluster centroid, whereas we may give lower weights to segments that are not mapped as well onto this unit. Furthermore, we may distribute the contribution of a segment being mapped onto a specific unit also across units in the neighborhood, utilizing the topology-preserving characteristics of the *SOM*. This allows for a more stable representation of the segments distribution across the segment map. A Gaussian centered at the winner can thus be used to model the contribution of each segment’s location onto the neighboring units, and thus onto the attributes of the feature vector for the music *SOM*, as indicated by the shadings in Figure 2.

We thus create a feature vector for each particular piece of music based on the distribution of its segments on the segment *SOM*. Training a second *SOM* using these feature vectors we obtain a clustering where each piece of music is mapped onto one single location on the resulting map, with similar pieces of music being mapped close to each other.

4 Experiments

For the following experiments we use a collection of 230 pieces of music, ranging from classical music, such as *Mozart’s “Kleine Nachtmusik”*, via some hits from the 1960’s such as *Cat Steven’s “Father and Son”* or *Queen’s “I want to break free”*, to modern titles, e.g. *Tom Jones’ “Sexbomb”*.

These songs were segmented into 5-second-intervals, of which every second segment was used for further processing, with a total of 17 frequency bands being

¹ We use the notation (x/y) to refer to the unit in column x , row y , starting with $(0/0)$ in the upper left corner

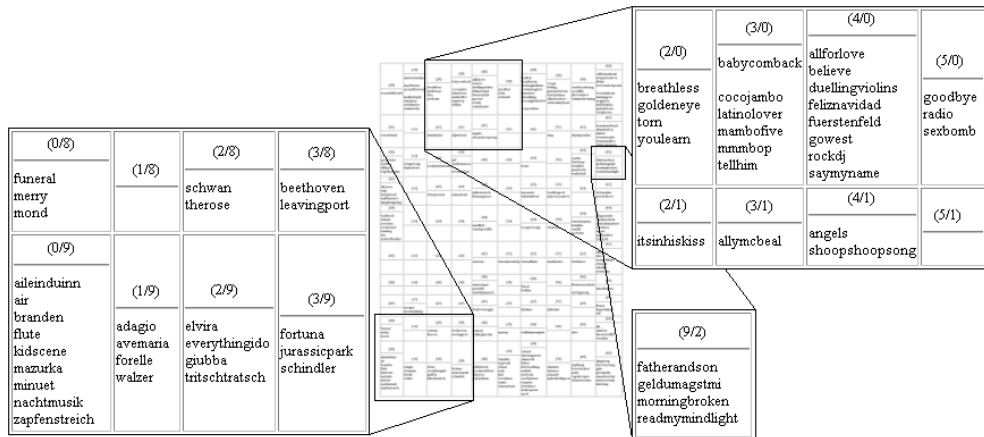


Fig. 3. SOM representing 230 pieces of music

selected. Following the Lagrange interpolations and FFT we thus end up with 5022 feature vectors representing the 5022 5-second segments of the 230 songs in a 4352-dimensional feature space. These feature vectors were further used to train a 22×22 dimensional segment SOM. Due to space restrictions we cannot provide a representation of the resulting map, yet we will use some examples for discussion.

For most songs the individual segments are mapped onto a rather small number of neighboring units. For example, we find most segments from classical titles mapped onto the lower right area of the segment SOM. To provide just one example of the segment SOM we find unit (13/21) to represent mostly classical segments mapped onto it, such as *Adagio*, *Air*, *Ave Maria*, *Beethoven's 5th Symphony*, *Brandenburg Concerts*, *Kleine Nachtmusik*, *Nocturne*, and many more. However, we also find the third segment of *Crash Boom Bang* by *Roxette* on this unit, which is definitely not a classical piece of music. Yet in this particular segment we find the music to be rather calm and "classic-like", resulting in the mapping of this particular segment onto this cluster. Furthermore, we find many intros as well as fade-out passages of songs to be mapped into the classic cluster on the segment SOM, as in those passages we quite frequently find violin or guitar music, independent of the instrumentation of the rest of the song. Some titles, such as *"Ironic"* by *Alanis Morissette* contain both rather soft and very dynamic passages and thus have their segments spread across several clusters co-located with segments from other songs of similar characteristics. However, the characteristics of some songs are too diverse to allow precise mapping of their segments and are thus spread across larger areas on the segment map.

In order to obtain a more compact representation of the musical archive, we create new feature vectors for each song based on the location of their segments. This results in a $22 \cdot 22$, i.e. 484-dimensional feature vector for each of the 230 songs. These feature vectors are used to train the 10×10 SOM represented in Figure 3.

Each song is now mapped onto one single position according to its musical characteristics. Taking a look at the Classic cluster in the lower left corner of the map,

we find unit (0/8) to represent the *Funeral March (Chopin)* as well as the *Moonlight Sonata (Beethoven)* and the *Merry Peasant (Schuhmann)*. All three pieces consist of rather calm piano music, and have their segments mapped mostly in the classic cluster on the segment map. Unit (0/9) also represents almost solely classical music, such as *Air* or the *Brandenburg Concerts* by *Bach*, as well as, again, pieces by *Schuhmann*, *Chopin* and *Mozart* (*Fremde Länder und Menschen (Schuhmann)*, *Mazurka (Chopin)*, and *Kleine Nachtmusik (Mozart)*). The song *Ailein Duinn* is a Scottish folk song and thus at first glance does not seem to fit into this cluster. Yet, when listening to it we find it to be a ballad sung by a woman and accompanied only by violin and harp, thus making it sound-wise fit perfectly into the classic cluster, even though it would be organized into the folk cluster by strict musical genre. On the neighboring unit (1/9) we find two pieces by *Schubert*, namely *Ave Maria*, *Themen und Variationen der Forelle*, as well as a piece by *Mozart*, i.e. the *Adagio* of his *Clarinet Concert*.

It is important to note, that the *SOM* does not organize the songs according to their melody, but rather according to their sound characteristics. If we move on to unit (2/9) we find it to represent some more dynamic pieces of classic music, such as the *Tritsch Tratsch Polka* by *Strauß* or again compositions by *Mozart*. We also find two vocal pieces on this unit, namely *Vesti la Giubba (Domingo)* and *Everything I do* by *Bryan Adams*. Obviously, *Everything I do* is not a classical piece of music in the strict sense, yet in terms of its music we find it very calm and to be accompanied by an orchestra most of the time. Even more interesting, we find unit (3/9) further to the right, and still more dynamic, to represent two pieces by *John Williams*, i.e. the main theme of the movies *Jurassic Park* and *Schindler's List*. We also find mapped onto the same unit both *Tchaikovsky's "Schwanensee"* as well as *Bette Midler's "The Rose"*, a very soft love song with mostly piano and violin passages, both on unit (2/8). This cluster also nicely demonstrates the topology preservation capabilities of the *SOM*, with the dynamics and intensity of the various pieces of music increasing from left to the right.

To pick just one further example from a different section of the map, we find *Cher's "Believe"*, *Robbie Williams' "Rock DJ"*, *The Pet Shop Boys' "Go West"* mapped together on unit (4/0) next to *Lou Bega's "Mambo No. 5"* and *Tom Jones' "Sexbomb"* on units (3/0), and (5/0), respectively. Another special mapping worth mentioning in this cluster is the co-location of three songs by a single singer. *Ally McBeal - Searching My Soul, Tell Him* and *It's in his Kiss* on the neighboring units (2/1),(3/0) and (3/1) respectively, are all sung by *Vonda Shepard* and were taken from the same CD, played by the same group with the same set of accompanying instruments. Furthermore, the title *It's in his Kiss* is a cover version of *Cher's Shoop Shoop Song*, which is also located in this cluster on unit (4/1).

5 Conclusions

Access to digital libraries requires, apart from query-and-retrieval based approaches, a means to explore and browse the available collections in order to get an understanding of the information available. While many approaches exist for the explo-

ration of textual libraries, access to multimedia collections has mainly been limited to retrieving items based on queries, be it textual or acoustic-based queries. Exploration of multimedia collections has only found limited support, with content-based browsing facilities mostly relying on preceding manual categorization of the items.

In this paper we presented an approach to create a content-based organization of music archives. The acoustic characteristics of pieces of music are analyzed, and the *Self-Organizing Map*, an unsupervised neural network, is used to create a mapping according to their similarity. Following the extraction of frequency spectra, segments of 5 seconds length are clustered, such that segments with similar sound characteristics are mapped physically close together on the resulting segment *SOM*. In this first step, music segments are organized to obtain a fine-grained representation of segment-wise similarities, based upon which a clustering of the complete songs can be obtained. In a second level clustering process, the distribution across the segment *SOM* is used to create a feature vector for each piece of music, which is fed into another *Self-Organizing Map*. On the second level map we thus obtain an organization of an archive of music according to sound similarities. We find, for example, classical pieces of music to be grouped closely together, set well-apart from hard-rock or pop music. On the other hand, we find rather “soft” pop titles with mainly classical instruments mapped closer to the classical sector on the map.

The presented approach supports browsing and exploration of music archives by automatically grouping titles by sound characteristics, creating a kind of genre-based organization. It thus nicely combines with and complements more traditional interfaces to music archives, such as conventional database queries for music meta-data, or advanced retrieval methods based on, e.g., melody.

In order to further improve the quality of the clustering, additional features will be incorporated. Apart from mere frequency spectra and their dynamics we are currently investigating the addition of beat information. Furthermore, modeling psycho-acoustic features should provide a better separation according to the perceived similarity of sounds.

References

1. D. Bainbridge, C. Nevill-Manning, H. Witten, L. Smith, and R. McNab. Towards a digital library of popular music. In E. Fox and N. Rowe, editors, *Proc of the ACM Conf on Digital Libraries (ACMDL'99)*, pages 161–169, Berkeley, CA, August 11-14 1999. ACM. <http://www.acm.org/dl>.
2. P. Cosi, G. De Poli, and G. Lauzzana. Auditory modeling and self organizing neural networks for timbre classification. *Journal of New Music Research*, 1994.
3. M. Dittenbach, D. Merkl, and A. Rauber. The growing hierarchical self-organizing map. In S. Amari, C. L. Giles, M. Gori, and V. Puri, editors, *Proc of the Intern. Joint Conf on Neural Networks (IJCNN 2000)*, volume VI, pages 15 – 19, Como, Italy, July 24-27 2000. IEEE Computer Society. <http://www.ifs.tuwien.ac.at/ifs/research/publications.html>.
4. S. Dixon and E. Cambouropoulos. Beat tracking with musical knowledge. In W. Horn, editor, *Proc of the 14th European Conf on Artificial Intelligence*, pages 626–630, Amsterdam, Netherlands, 2000. IOS Press. <http://www.ai.univie.ac.at/~simon/>.
5. B. Feiten and S. Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3):53–65, 1994.

6. J. Foote. Content-based retrieval of music and audio. In C. Kuo, editor, *Proc of SPIE Multimedia Storage and Archiving Systems II*, volume 3229, pages 138–147, 1997. <http://www.fxpai.xerox.com/people/foote/papers/spie97-abs.html>.
7. A. Ghias, J. Logan, D. Chamberlin, and S. B.C. Query by humming: Musical information retrieval in an audio database. In *Proc of the Third ACM Intern. Conf on Multimedia*, pages 231–236, San Francisco, CA, November 5 - 9 1995. ACM. <http://www.acm.org/dl>.
8. M. Hawley. The personal orchestra. *Computing Systems*, 3(2):289–329, 1990.
9. T. Kohonen. *Self-organizing maps*. Springer-Verlag, Berlin, 1995.
10. T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self-organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, May 2000. <http://ieeexplore.ieee.org/>.
11. X. Lin. A self-organizing semantic map for information retrieval. In *Proc of the 14. Annual Intern. ACM SIGIR Conf on Research and Development in Information Retrieval (SIGIR91)*, pages 262–269, Chicago, IL, October 13 - 16 1991. ACM. <http://www.acm.org/dl>.
12. R. McNab, L. Smith, J. Witten, C. Henderson, and S. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proc of the 1st ACM Intern. Conf on Digital Libraries*, pages 11–18, Bethesda, MD, USA, March 20 - 23 1996. ACM. <http://www.acm.org/dl>.
13. M. (MMA). MIDI 1.0 Specification, V 96.1. online, March 1996. <http://www.midi.org>.
14. mp3. mp3.com. Website, May 2001. <http://www.mp3.com> as of May 2001.
15. Y. Muraoka and M. Goto. Real-time rhythm tracking for drumless audio signals - chord change detection for musical decisions. In *Proc of the IJCAI97 Workshop on Computational Auditory Scene Analysis*, Nagoya, Japan, August 23-29 1997. <http://www.etl.go.jp/goto/PROJ/bts-j.html>.
16. F. Nack and A. Lindsay. Everything you wanted to know about MPEG7 - part 1. *IEEE MultiMedia*, pages 65–77, July-September 1999.
17. S. Pfeiffer, S. Fischer, and W. Effelsber. Automatic audio content analysis. In *Proc of the Fourth ACM Conf on Multimedia*, pages 21–30, Boston, USA, November 18 - 22 1996. <http://www.acm.org/dl>.
18. A. Rauber, M. Dittenbach, and D. Merkl. Automatically detecting and organizing documents into topic hierarchies: A neural-network based approach to bookshelf creation and arrangement. In J. Borbinha and T. Baker, editors, *Proc of the 4. European Conf on Research and Advanced Technologies for Digital Libraries (ECDL2000)*, number 1923 in Lecture Notes in Computer Science, pages 348–351, Lisboa, Portugal, September 18-20 2000. Springer. <http://www.ifs.tuwien.ac.at/ifs/research/publications.html>.
19. A. Rauber and D. Merkl. The SOMLib Digital Library System. In S. Abiteboul and A. Vercoustre, editors, *Proc of the 3. European Conf on Research and Advanced Technology for Digital Libraries (ECDL99)*, number LNCS 1696 in Lecture Notes in Computer Science, pages 323–342, Paris, France, September 22-24 1999. Springer. <http://www.ifs.tuwien.ac.at/ifs/research/publications.html>.
20. A. Rauber and A. Müller-Kögler. Integrating automatic genre analysis into digital libraries. In *Proc of the First ACM-IEEE Joint Conf on Digital Libraries*, Roanoke, VA, June 24-28 2001. ACM.
21. Y. Tseng. Content-based retrieval for music collections. In *Proc of the 22. Annual Intern. ACM SIGIR Conf on Research and Development in Information Retrieval*, pages 176–182, Berkeley, CA, August 15 - 19 1999. ACM. <http://www.acm.org/dl>.
22. E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification search and retrieval of audio. *IEEE Multimedia*, 3(3):27–36, Fall 1996.

Andreas Rauber, Markus Frühwirth: **Automatically Analyzing and Organizing Music Archives.**

In: Proc of the 5. European Conf on Research and Advanced Technology for Digital Libraries (ECDL 2001), Sept. 4-8 2001, Darmstadt, Germany, Lecture Notes in Computer Science LNCS, Springer, Germany.