One-Touch Access to Music on Mobile Devices

Dominik Schnitzer[1,2], Tim Pohle[1], Peter Knees[1], and Gerhard Widmer[1,2]
[1]Department of Computational Perception, Johannes Kepler University Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence (OFAI)
dominik.schnitzer@jku.at

ABSTRACT

We present an approach that offers the user a convenient and meaningful way to access her music on a mobile device. By exploiting information on acoustic similarity and community-based music labels, a music collection is automatically structured and described to allow for easy orientation and navigation within the collection. To this end, the complete collection is arranged along a circular playlist path such that similar sounding pieces are grouped together. As a consequence, regions of musical styles emerge. Furthermore, we propose two approaches to derive informative descriptors that are displayed on the different regions, allowing an overview of the whole collection at a glance. For demonstration, we implemented our prototype interface on an Apple iPod.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Selection process*

General Terms

Algorithms, Design

Keywords

Mobile Music Information Retrieval, community-based artist clustering, automatic music collection organisation

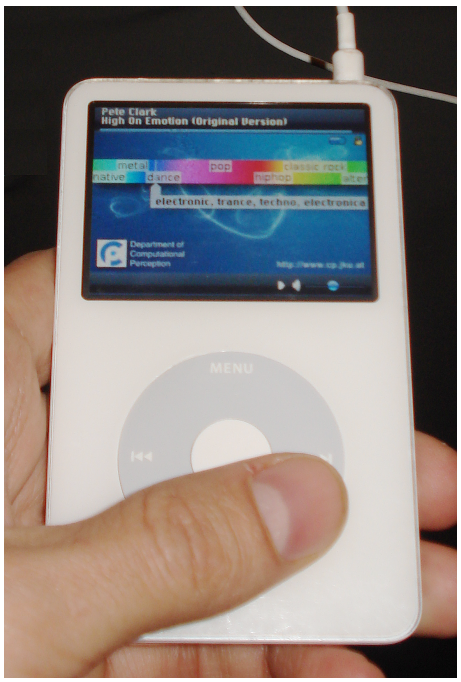

Figure 1: Implementation of our interface prototype on an Apple iPod.

1. INTRODUCTION

Thanks to advancements in information technology, the amount of music that can be stored on portable players keeps on growing. However, with more music available, the question of how to conveniently access the music en-route gets more and more important. Looking at the way music is accessed today on mobile players, it gets apparent that current techniques still leave space for improvements. State-of-the-art interaction is either based on random shuffling (resulting in very inconsistent, and thus unfitting playlists) or on *artist–album* categorisations. As a consequence of the latter, the user needs to know exactly which kind of music she wants to listen to in advance. Furthermore, such a navigation concept makes it necessary to pay relatively much attention to the player during the selection process, which is not desirable in many situations.

In [14, 12], we proposed an approach that addresses some of these problems. The content of a music collection is analysed and arranged along a circular path ("The Wheel") such that similar sounding music is grouped together. Thus, the complete music collection can be accessed by simply turning a knob. As the collection is sorted according to acoustic similarity, also subsequently played pieces, i.e. the playlist, fit together automatically. However, this approach has at least two disadvantages: First, the user has to learn from experience where to expect which kind of music, and sec-

MUM'07, December 12–14, 2007, Oulu, Finland.

ond, due to the applied techniques, very similar music could possibly be found in different regions.

In this work, we tackle these limitations by enhancing the purely audio-based approach from [14, 12] with music-related community data. Using this community data, we can not only improve the consistency of music piece grouping along the wheel, but also advance the user interface such that it provides an intuitive overview over the complete collection at a glance. This is accomplished by automatically labeling the different regions of the wheel with musically meaningful tags. In addition, we present an implementation of the resulting interface on an Apple iPod (cf. Figure 1).

The remainder of this paper is organised as follows: In the next section, we sketch the basic idea of our approach and review related work. In Section 3, we describe the techniques underlying our proposed approach. In Section 4, we evaluate our techniques on a small collection of 1,440 tracks. Finally, we report on implementation details of our prototype interface on an Apple iPod in Section 5.

2. BASIC IDEA AND RELATED WORK

As stated above, the idea behind our interface is to offer the user a convenient and meaningful way to access her music on a mobile device. To this end, we adopt the metaphor of "The Wheel" from [14, 12]. In this approach, each position on a circular (playlist) path is associated with a track in the collection by applying a *Traveling Salesman Problem (TSP)* algorithm to the collection's full audio distance matrix. This matrix is obtained by building a statistical model for each music piece and calculating pairwise distances between all models. As a result of the TSP, similar tracks, i.e. tracks with a small distance, are grouped together, so that certain regions of the wheel can be associated certain musical styles. This allows the user to select the kind of music she wants to listen to by simply turning the wheel into a certain region.

In [12, 5], we also presented an approach to improve the quality of the generated playlists by incorporating Web-based data. Using the artist information found in the mp3 files' *ID3 tags*, queries are sent to Google to obtain related Web pages. Based on the retrieved Web pages, a $tf{\times}idf$ vector is calculated for each artist. In the next step, these artist vectors are clustered using a *Self-organising Map (SOM)*. The resulting map is used to determine which artists can be considered to be similar. This definition of artist similarity is then used to modify the audio distance matrix, in such a manner that the distances of pieces of artists that are not similar (according to the clustering) are artificially increased. Thus, for the TSP algorithm, it is less attractive to select transitions between tracks of dissimilar artists. We found indication that this modification leads to less noisy playlists, i.e. playlists that do not switch between genres as frequently as playlists created from audio information only. However, while this enhancement leads to better playlists in terms of genre consistency, it does not improve the quality of the user interface. Still, the user has to learn from experience which regions on the wheel represent which kind of music. Thus, it may require some time to find music from a momentarily desired style.

In contrast, in this paper, we present an enhanced approach that allows for intuitive navigation and orientation within large music collections by displaying meaningful descriptors along the playlist. Based on the descriptions, the user can easily find music she wants to listen to by browsing the complete collection with one touch. The data necessary to obtain the displayed descriptors is retrieved from last.fm/Audioscrobbler. One major advantage of this data source over the Google-based approach from [12, 5] is that for each contained artist only one Web request has to be made instead of 51. Furthermore, as the last.fm community data can be used to perform a clustering of artists prior to the playlist generation step, our approach avoids calculation of a TSP on the complete collection.

Finally, we want to give an overview over other related work. In [9], Self-organising Maps are utilized for browsing of music collections and intuitive playlist generation on portable devices. Having the music pieces arranged on a 2-dimensional map (where similar sounding pieces are grouped together), the user can simply create playlists by drawing arbitrary paths on the map with a pen. In [4], Baumann et al. put a focus on socio-cultural settings and different modalities of song similarity. For ease of use, they present an implementation of a music recommender on a PDA, where multiple dimensions of similarity can be accessed via a joystick. In [17], the similarity of artists is visualised on mobile devices by means of graph-drawing algorithms.

Beside other approaches to automatic playlist generation (e.g. [1, 3, 7]), especially methods that allow for *dynamic playlist generation* [11, 16] are of interest in regard to this paper, as they are particularly suited for implementation on mobile devices. The basic idea is to incorporate explicit feedback given by the user, more precisely, rejections of songs ("skips"). Based on the skips performed, the rest of the playlist is modified to (presumably) better match the user's expectations. In future work, these approaches could be used to complement the interface we present in this paper.

3. PROPOSED METHOD

In this section, we briefly describe the techniques used to build our interface, namely retrieval of artist-related community data, calculation of audio similarity between individual music pieces, and construction and labeling of a circular playlist.

3.1 Last.fm-based Artist Clustering

In a first step, we aim at obtaining descriptors from last.fm for all artists in the collection. Last.fm[1] is a music information service that integrates into music player software and keeps track of each user's listening habits. Additionally, the users can tag their tracks and artists with arbitrary labels. Using the information from the tags together with knowledge on users' collections, services like personalized radio stations or recommendation of similar music are provided. One aspect that is highly valuable for research and other applications is that large parts of the collected data are made available via a Web service[2].

For a specified artist, the tag data takes the form of a list of words with associated (normalised) weights. Based on this information, we cluster artists into (e.g., five) main clusters. Tracks with unknown artist as well as tracks from artists that can not be found on last.fm, are put into the best-matching cluster based on audio similarity (cf. Section 3.2). Using this procedure, each cluster is also as-

[1] `http://www.last.fm`

[2] `http://www.audioscrobbler.net`

signed a label that describes what is common to the contained artists. We present two approaches for this clustering/labeling step, as outlined next.

3.1.1 Tag Frequency Binning

A simple method to group artists can be easily implemented using the acquired data: In a first step, each artist is joined with artists sharing the same top ranked last.fm tag. Besides this initial assignment, each artist is also weighted by the number of associated songs in the collection. That way, weighted tag-bins of artists are formed. In a next step, the bin with the smallest weight is removed and its artists are spread among the remaining bins according to their second highest ranked tag. After that, again, the bin with the smallest weight is broken up. This procedure is repeated iteratively until the desired number of groups is found.

An obvious drawback of this simple technique is that not all artists can be directly related to one of the main clusters since some artists are not assigned one of the top tags at all. The tracks by those artists can again be associated with a group using music similarity (cf. Section 3.2). Furthermore, this method can only work well when preprocessed, normalized and weighted tags (like the tags from last.fm) are available. In the next section, we present a more general approach to generating the initial grouping.

3.1.2 NMF-Based Approach

As some of the artist tags consist of more than a single word and some of these words also appear in various flections (e.g., *b-boy* and *b-boying*, or *oldie* and *oldies*), we chose to break all tags into unigrams and apply stemming. Each of the newly obtained unigrams inherits the weight of the original last.fm tag. Weights of tags that appear more than once for one artist are summed up.

The resulting data is analyzed for common "topics" (in our case, musical concepts) using *Non-Negative Matrix Factorization (NMF)* [6] (cf. [18, 13]). This is accomplished by constructing a matrix of size *number of artists* $\times$ *number of terms* containing the weights of each term for each artist. The resulting decomposition yields an a-priori defined number of topics. The initial artist descriptors can then be transformed into a representation that reflects the degree of relatedness to each topic. We observe that in most cases, each artist is mainly associated with one of these topics. Thus, we simply assign each artist to the topic/cluster it is most related to.

3.2 Audio Similarity

For calculating music similarity based on the audio content, we apply the algorithm from [15], which is a modified version of the algorithm proposed in [10]. Pairwise distances between all music tracks in the collection are determined. For each piece of music, *Mel Frequency Cepstral Coefficients (MFCCs)* are computed on short-time audio frames to characterize the frequency distribution of each frame and hence model aspects of timbre. Following [2], the i-th MFCC is defined as:

$$c_i = \frac{1}{2\pi} \times \int_{w=-\pi}^{\omega=+\pi} \log(S(e^{j\omega})) \cdot e^{j\omega \cdot n} \partial\omega \qquad (1)$$

On each frame, 25 MFCCs are computed. Ignoring the temporal order of frames, each song is then represented as a *Gaussian Mixture Model (GMM)* of the distribution of MFCCs. Using a Single Gaussian Model with full covariance matrix is sufficient for representation, facilitating computation and comparison of the models [8]. A distance measure is calculated on these models, denoted by D_{Gauss}.

To complement the MFCC-based similarity component, also *Fluctuation Patterns (FPs)* are computed. This can be considered an abstract description of the temporal succession of frequency activations. A track is represented as a 12-band spectrogram and for each band, a *Fast Fourier Transformation (FFT)* of the amplitude is taken over a window of six seconds. The resulting matrix of size *number of bands* $\times$ *number of FFT components* describes one song and is referred to as the Fluctuation Pattern of the song. The FPs of two songs are compared by calculating the cosine similarity between them, denoted by D_{FP}. Furthermore, two additional FP-related features are computed: *Bass (FPB)* and *Gravity (FPG)*. These two features are scalar and the distance between two songs is calculated by subtracting them, denoted by D_{FPB} and D_{FPG}. To obtain an overall similarity value measuring the similarity of two songs, all described similarity measures are z-normalized and then combined by a simple arithmetic weighting.

$$D = 0.7 \cdot D^{Norm}_{Gauss} + 0.1 \cdot \left(D^{Norm}_{FP} + D^{Norm}_{FPB} + D^{Norm}_{FPG} \right) \qquad (2)$$

where D^{Norm}_x is the value of D_x after normalization. Details can be found in [10, 15]. The similarity measure is used to determine the order in which tracks are arranged within a cluster, as described in the next section, and also to assign tracks with no last.fm data available to one of the top-level clusters.

3.3 Playlist Generation

Once the top-level clusters are defined, and the association of each track to a particular cluster has been accomplished, two steps need to be performed in order to create a circular arrangement of the collection. First, the placement of individual tracks within a particular cluster must be determined. Second, the different clusters have to be connected.

3.3.1 Arranging Tracks in a Cluster via TSP

All tracks within a region should be arranged based on their audio similarity to obtain a smooth transition between consecutive tracks and subregions of similar tracks within a given cluster. In [14, 12], a Traveling Salesman Problem (TSP) algorithm is applied to organise a given music collection. Here, we adopt this procedure. The "cities" to visit are the tracks, while distances between the tracks are calculated with the audio-based music distance function described in Section 3.2. We use the *Minimum Spanning Tree (MST)*-based algorithm as proposed in [12]. From the distance matrix of the tracks in a cluster, an MST is created. The tree is traversed in a depth-first order, and the tracks are written into a list in the order they are first visited. Finally, the first track is visited again to form a closed route. Thus, all tracks are ordered into a circular playlist within each cluster.

3.3.2 Connecting the Intra-Cluster Playlists

Having a circular playlist for the tracks within each top-level cluster we are in need of a strategy to optimally connect them. The idea is to find an arrangement of clusters that

reflects the similarity of the contained music. Furthermore, we need to determine those tracks that are best suited as links between the clusters.

We decided to use a greedy approach: The output of the previous step is n circular routes, where n is the number of clusters. To merge them, each of the n circular playlists is broken at the largest "distance" (i.e., the two consecutive tracks in the playlist with the largest audio distance between them is searched). These two tracks become the endpoints of the (now linear) playlist. To connect these n linear playlists, the two most similar endpoints (according to audio similarity) from different clusters are connected, yielding $n-1$ linear playlists. This procedure is repeated until all intra-cluster playlists are merged to one long circular playlist containing all tracks in the collection.

3.4 Automatic Region Labeling

To support a more easy orientation within the music collection, the playlist created in the preceding steps is visualised and augmented with meaningful descriptors. For this purpose, we align the whole playlist along a sliding bar that is colored according to the distribution of the previously identified clusters. Thus, different kinds of music are represented as different colors on the bar. Between clusters, colors are faded to support the impression of smooth transitions. Furthermore, the labels of the top-level clusters are displayed in the center of the corresponding cluster region on the color bar. While browsing, also additional tags that describe the currently viewed track are displayed. This allows for an immediate overview of the complete collection.

4. EVALUATION

In order to evaluate the usefulness of combining last.fm data with content-based audio similarity, we compiled a small test collection containing 1,440 tracks by 959 artists from 21 genres. Last.fm tags were available for 834 artists (87%). Using the simple *Tag Frequency Binning* approach, 135 artists could not be assigned to one of the top-level clusters, i.e. in total, only 73% of the artists could be preclustered. As a result, 1,128 tracks (78%) can be assigned to a cluster without any information on the audio similarity. Using the NMF-based clustering approach, this can be accomplished for 1,299 tracks (90%).

To gain insights into the effects of community-based preclustering and audio similarity-based arrangement within the clusters we investigate the consistency of the generated playlists (according to the genre information). Figure 2 depicts the distribution of the 21 genres along the playlists for three different approaches — NMF pre-clustering with random placement of tracks within the clusters (top), NMF preclustering with TSP-based track placement (middle), and Tag Frequency Binning pre-clustering with TSP-based track placement (bottom). Dark segments indicate a high agglomeration of tracks from one genre. The genres are ordered by the index where most pieces of that genre are accumulated. (In the optimum case, a playlist would thus tend to result in a diagonally descending sequence of black bars.)

One can see that already the community-based pre-clustering alone has a very positive impact on the structure of the playlist. Incorporating audio similarity information by applying a TSP algorithm to the tracks in the clusters improves the consistency even further. This impression can be confirmed by calculating the entropy of the genre distribution for each approach. To this end, we investigate short sequences of the playlists and count how many of t consecutive tracks belong to each genre. The normalised result is then interpreted as a probability distribution on which the Shannon entropy can be calculated as

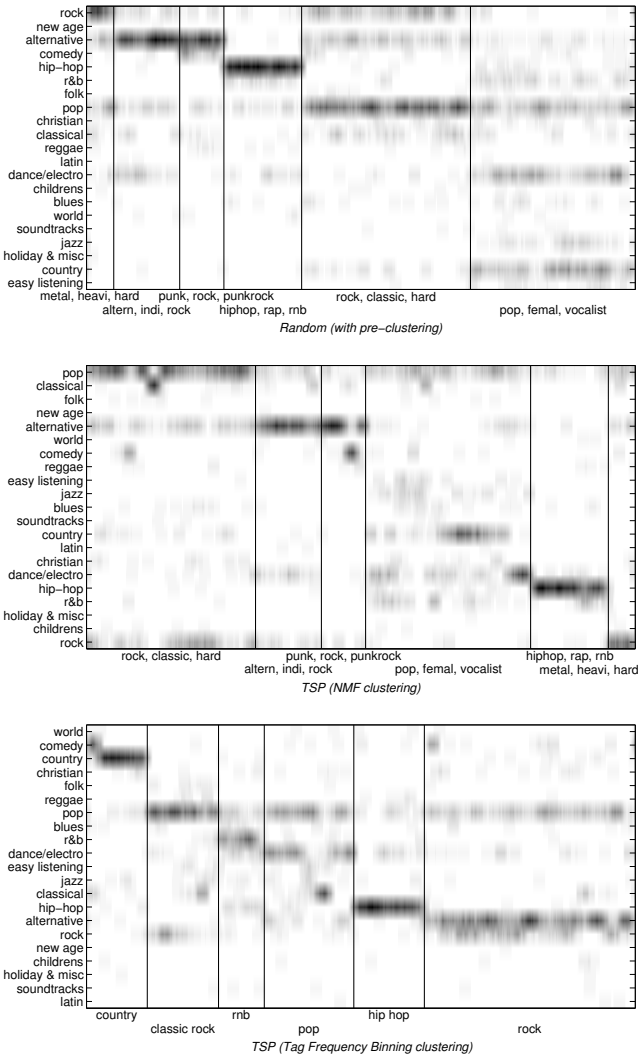

Figure 2: Genre distributions in playlists generated on our test collection with three different approaches. Vertical lines represent top-level cluster boundaries.

$$H(X) = -\sum_{x} p(x) \log_2 p(x) \tag{3}$$

where $p(x)$ is the relative frequency of genre x and $\log_2 p(x) = 0$ if $p(x) = 0$.

In Figure 3, the entropy values for $t = 2...12$ averaged over the whole playlist are given, i.e. each track of the playlist was chosen once as the starting track for a sequence of length t. Obviously, both audio supported playlist generation methods perform better than the random distribution of tracks within a top-level cluster. The two different clustering approaches yield comparable results.

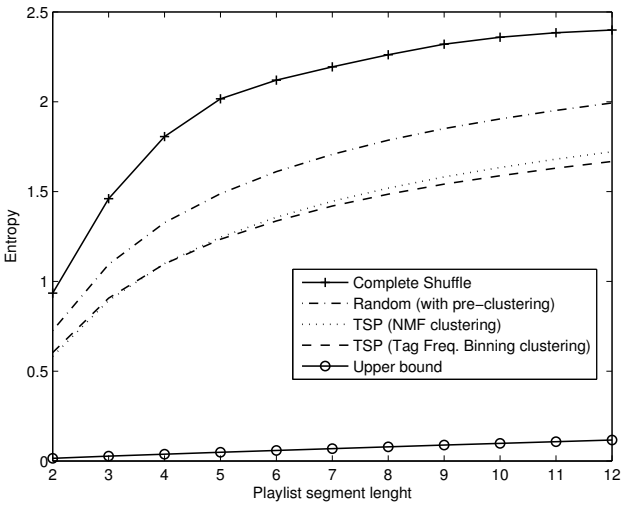

Figure 3: Average entropy values for short sequences of the playlists. (Lower values are better.)

5. PROTOTYPE IMPLEMENTATION

To demonstrate and test how the proposed interface effectively works, we brought the whole idea to life on an Apple iPod Video. The iPod platform was selected as its scroll-wheel offers an intuitive input device for navigation and thus fits well for the browsing interface presented here.

As the iPod is a closed platform, developing plugins or modifying parts of the player-application is not possible. So it was necessary to use an open, third party, development environment to prototype the interface. We selected Rockbox[3], a GPL-licensed Open-Source DAP firmware.[4]

5.1 The User Interface

Our implementation of the music browsing interface directly integrates in the main mobile audio playback application and makes easy changing of music style possible, even during playback. Figure 4 shows the finished prototype implementation and describes its main elements.

To understand how the player is used, the main focus should be put on the continuous color bar in the center of the screen. The color bar shows the six main music labels which were found to describe the collection. The labels are weighted and color-coded according to Section 3.4. As such, the color bar alone allows a quick and coarse first glance on the music collection currently loaded. For clarity the number of labels displayed is limited to a maximum of six.

In the original Rockbox and Apple iPod player the scroll wheel is mainly used for controlling the playback volume. For our purpose we remapped the input of the scroll wheel to allow circular movement of the continuous color bar to select a specific area to play. A static pin overlaying the bar indicates the currently selected area. In addition to this, a textfield displays last.fm subtags which were found to describe the selected region.

[3]`http://www.rockbox.org`

[4]The modified Rockbox implementation and player application skin are available on the project homepage at `http://www.cp.jku.at/projects/intelligent-ipod/`

Figure 4: The different elements of the browsing interface. A continuous color bar shows six main music styles discovered for a collection of 1,440 songs (2). The bar can be slid to the left or to the right using the scroll wheel (4). Additional information about a subregion is displayed in a textfield (1) and updated while moving the color bar. To start playing the music of a specific area, one can simply scroll to that area and press the select button (5). The currently played song is displayed in (3).

As soon as the user selects a position on the color bar for playback, the selection is extrapolated on a song in the precomputed TSP playlist (see Section 3.3). Since all songs in this playlist are arranged according to their maximum audio similarity, successive songs should fit together nicely, making the interface perfect for quick but matching on-the-go playlists.

5.2 Usage Scenario

To use the proposed interface on the iPod, three items need to be generated and uploaded on the mobile audio player: First, a playlist, generated by the means of the arrangement algorithm presented in Section 3.3. Second, a text file, containing the region and sub-region labels of the playlists extracted from last.fm, and third, a pre-generated color bar image, as described in Section 3.4.

Since it is computationally very intensive to compute optimal playlist arrangements, analyze audio content and process huge loads of last.fm tags, these computations can not be performed on a mobile device. An actual usage scenario would definitely have to include a desktop or server computer, synchronizing the required data to the mobile player.

6. DISCUSSION AND FUTURE WORK

We presented an intelligent mobile music interface that allows for intuitive navigation and orientation within large music collections. By visualising the different music styles via different colors and displaying meaningful descriptors along the generated playlist, the user can easily find music she wants to listen to with one touch.

In general, feedback from people we asked to play around with the interface was very positive. People were happily surprised by the simple possibility to explore and listen to a music collection by just scrolling and clicking. Also the arrangement of tracks has been mentioned to be very fitting. The easy selection process of musical styles was considered to be very practical when being en-route (although not actually tried in a real-life scenario). Regarding the two different clustering approaches proposed, although the discovered concepts were very similar, we had the impression that the tag frequency binning approach resulted in slightly more intuitive clusters.

As for future work, as mentioned before, the incorporation of *dynamic playlist generation* based on skipping behaviour (cf. [11]) seems to be very promising. Currently, the collection is organised in a static way which could be tedious after some time. Hence, taking user feedback, i.e. skipping of currently undesired tracks, into consideration would make the listening experience more interactive and interesting.

In addition, we plan to investigate further options to enable a more detailed selection of tracks within the music collection. To this end, we will explore methods that provide position-based level-of-detail adaptation, e.g. an automatic magnification mode. We are also confident that the simplicity of our approach allows us to easily transfer the interface to other mobile platforms such as the Nokia S60 series or the upcoming iPod Touch. In general, we believe that the capabilities of such platforms will offer a playground for new and cool approaches to interact with music.

7. ACKNOWLEDGMENTS

This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number L112-N04. The Austrian Research Institute for Artificial Intelligence acknowledges financial support by the Austrian ministries BMBWK and BMVIT.

8. REFERENCES

[1] M. Alghoniemy and A. Tewfik. A Network Flow Model for Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'01)*, Tokyo, Japan, 2001.

[2] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, Paris, France, 2002.

[3] J.-J. Aucouturier and F. Pachet. Scaling Up Music Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'02)*, Lausanne, Switzerland, 2002.

[4] S. Baumann, T. Pohle, and V. Shankar. Towards a socio-cultural compatibility of MIR systems. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004.

[5] P. Knees, T. Pohle, M. Schedl, and G. Widmer. Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation. In *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'06)*, Santa Barbara, CA, USA, 2006.

[6] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788791, 1999.

[7] B. Logan. Content-Based Playlist Generation: Exploratory Experiments. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, Paris, France, 2002.

[8] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[9] R. Neumayer, M. Dittenbach, and A. Rauber. PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[10] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Austria, March 2006.

[11] E. Pampalk, T. Pohle, and G. Widmer. Dynamic Playlist Generation Based on Skipping Behaviour. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[12] T. Pohle, P. Knees, M. Schedl, E. Pampalk, and G. Widmer. "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia*, 9(3):567–575, 2007.

[13] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Building an interactive next-generation artist recommender based on automatically derived high-level concepts. In *Proceedings of the 5th International Workshop on Content Based Multimedia Indexing (CBMI 2007)*, Bordeaux, France, 2007.

[14] T. Pohle, E. Pampalk, and G. Widmer. Generating similarity-based playlists using traveling salesman algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, 2005.

[15] T. Pohle and D. Schnitzer. Striving for an Improved Audio Similarity Measure. In *4th Annual Music Information Retrieval Evaluation Exchange*, 2007.

[16] D. Schnitzer. MIRAGE – High-Performance Music Similarity Computation and Automatic Playlist Generation. Master's thesis, Vienna University of Technology, 2007.

[17] R. van Gulik, F. Vignoli, and H. van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004.

[18] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th ACM SIGIR*, Toronto, Canada, 2003.