Innovative User Interfaces for Accessing Music Libraries on Mobile Devices

A SOM Based Music Browser for Mobile Devices

Peter Hlavac

Department of Software Technology, Vienna University of Technology
Favoritenstrasse 9-11 / 188, A-1040 Vienna, Austria
e9625925@stud3.tuwien.ac.at,
http://www.ifs.tuwien.ac.at/mir/

Abstract. A music browser *MobileSOM* with a new intuitive user interface designed for mobile devices with a built-in touch screen is presented. Music titles can be browsed, previewed and added to playlists picked from a Self-Organizing Map (SOM), on which similar pieces of music are grouped together. The songs from the created and editable playlists can be played on mobiles or streamed to an external player. The software, which is entirely programmed in J2ME, has been successfully tested on emulators and hardware devices by various manufactures that support the Java Multimedia API (MMAPI). A comparison between those mobiles in an emulator- and a real world environment is given.

Keywords: SOM, J2ME, MMAPI, Music, Mobile, Browsing

1 Introduction

The number of music portals that offer digital content to download, has grown rapidly within the last years. Thanks to the growing popularity of 3G networks the mobile market will be a main distribution channel not only for ring tones but as well for full track downloads. Increasing bandwidth and lower data transferring costs encourage users to access digital content over their mobile phones. Most of today's mobile devices have fewer resources compared to average desktop systems. User interfaces and program logics have to be adapted or even redesigned for those devices. The focus of the work presented in this paper is on graphical user interfaces and navigation opportunities that differ from the typical database-oriented access paradigm. MobileSOM (Fig. 1) offers an intuitive user interface that enables a simple way of selecting music titles from a graphical map on which similar pieces of music are grouped together. In this manner generated playlists can then be played on mobile devices or streamed to an external player. The remainder of this paper is organized as follows: Section 2 provides a view of how the work is positioned in the context of music information retrieval and used visualisation techniques. Sections 3 describes the workflow starting from extracting features from audio signals to the step where

Fig. 1. MobileSOM on a Nokia 7710 Device

pre-processed data is ready to be used via the MobileSOM software. In Section 4 generating playlists on mobile devices is discussed. Section 5 takes a closer look at the software itself, describes its features and presents experiences on running the software on emulators and real world devices. In the last section conclusions as well as an outlook on future work is given.

2 Related Work

Origin of this work is the project SOM-enhanced JukeBox (SOMeJB) [9], developed by the Institute of Software Technology and Interactive Systems at the Vienna University of Technology. The SOMeJB is an approach to automatically create an organization of a music archive following the sound similarity of the music files. More specifically, characteristics of frequency spectra are extracted and transformed according to psychoacoustic models. Resulting psychoacoustic rhythm patterns are further organized using an unsupervised neural network, the Growing Hierarchical Self-Organizing Map [7]. Particularly, a Self-Organizing Map is employed 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, leaving a user with related music next to it, allows intuitive exploration of a music archive. Different visualization techniques can be found in Islands of Music (IoM) [5]. The PlaySOM [2] and PocketSOMPlayer [4] applications both enable users to browse collections, select tracks, export playlists as well as listen to the selected songs. The PlaySOM presents a rich interface, offering different selection models, a range of visualizations, advanced playlist refinement, export functionality to external player devices and a basic playback mode of selected songs. The PocketSOMPlayer, on the other hand, offers a slim version of the desktop application, optimized for mobile devices. *MobileSOM* is an enhanced version of the PocketSOMPlayer with more sophisticated features and is the first SOM based player entirely programmed in J2ME.

3 Pre-processing Content

In order to run MobileSOM the location of the music content and how the songs are related to each other regarding their similarities must be defined first. The latter is done in the SOMeJB [9] where audio files are analysed and processed in a way that a computer is able to distinguish between different kinds of music. Features like timbre, loudness and rhythm are extracted by computing the power spectrum of the audio signal to obtain loudness values for specified frequency bands over a certain time period. For a detailed description on feature extraction see [8]. Those features are used to describe a piece of music in a mathematical representation and are stored in a feature vector. The more similar two pieces of music regarding their extracted features are, the smaller the (Euclidian) distance between their vectors is. Using those vectors the *Self-Organizing Map* (SOM) [3] algorithm is applied in PlaySOM [2] to organize the music files on a two-dimensional map display in such a way that similar pieces are grouped close together. The algorithm works as follows: The SOM consists of a set of units, which are arranged on a two-dimensional grid. Each of the units is assigned to a randomly initialized model vector that has the same dimension as the feature vectors. In each learning step a randomly selected feature vector is matched with the closest model vector (winner). An adaptation of the model vector is performed by moving the model vector closer to the feature vector. The neighbours of the winner are adapted as well but not as far as the model vector of the winning unit. This enables a spatial arrangement of the feature vectors such that alike vectors are mapped onto regions close to each other in the grid of the units. Once the learning phase is completed, each feature vector is mapped to its closest unit. In the next step the smoothed data histogram (SDH) [6] algorithm is applied to graphically visualise the SOM. Units are coloured regarding to the amount of songs per unit, starting from white (high density) over yellow to green (lower density), which can be interpreted as mountains, where as units with less assigned vectors are painted blue, which is a metaphor for water. Finally the visualisation of the SOM and the mapping between all titles with their corresponding units are exported to a jpg and a data file, which are used as input for the MobileSOM. The SOM consists of 280 units arranged with 20 units per row. The resolution of the jpg is 200x140 pixels. Hence every unit belongs to a square with dimensions 10x10 pixels on the map. Figure 2 illustrates all pre-processing stages at a glance.

4 Playlist Generation on Mobile Devices

Traditional ways of generating playlists like dragging and dropping pieces of music from a file explorer in to a player need not be suitable for mobile devices. Browsing through large directory structures on devices with small displays is a complex and inconvenient task. If a user is searching for a specific song he might want to find the piece of music by typing in the title or artist name which is, because of the lack of a keyboard on most devices, again a burdensome task. The

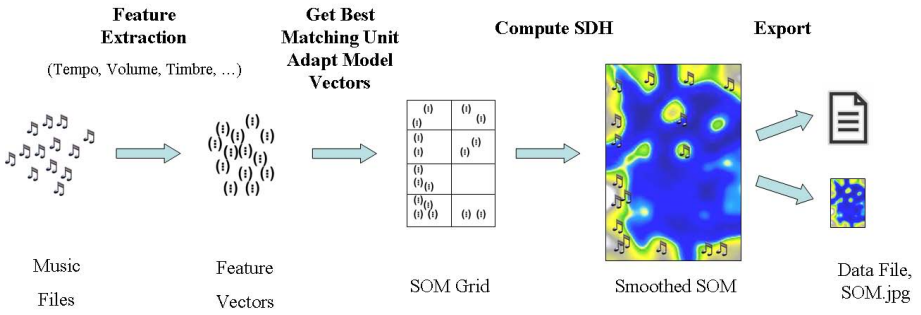

Fig. 2. Pre-processing Content

idea of displaying a user's entire music collection as a single map on his device opens new possibilities in creating playlists more intuitively and efficiently. Like described in the previous section the map is structured that similar pieces of music are grouped together. Song titles are displayed on the screen by pointing on a specific location on the map, so the user is able to get a feeling where different styles of music are located. After familiarizing with the SOM, the user can change from the preview into the playlist creation mode. By drawing a path or pointing on a single position on the map, one or more songs are selected and added to the playlist. Now just after view clicks and movements a playlist was generated quickly with the following advantages. Playlists generated in that way always contain different content, because it is hard to ever draw extactly the same path again. Users will not get bored in listening to the same songs in the same order. The playlists are more homogen than random generated ones because of music similarity. So the user might create different path for different moods. Accordingly the list can be altered by removing or changing the position of music titles in the edit mode. The items from the final playlist can be played back from the device, streamed from a server or send to a server. In the latter case the mobile device acts as a remote control. (See section 5.3) Figure 3 illustrates the stages for generating a playlist at a glance.

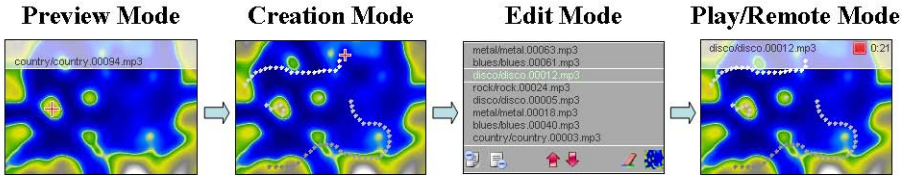

Fig. 3. Player Modes

5 MobileSOM

In this section a detailed description about the implementation and the key features of the MobileSOM, which is a prototype that demonstrates the interactive and intuitive generation of playlists on mobile devices, is given. The software was programmed in Java 2 Platform, Micro Edition (J2ME) which is a collection of Java APIs for the development of software for resource constrained devices such as PDAs and cell phones. Every Java Micro Edition device implements a profile. A profile is a superset of a configuration, of which there are currently two: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). The CLDC contains a strict subset of the Java class libraries, which are necessary for a Java virtual machine to operate. From version CLDC1.1 onwards floating point support is included which is a requirement in order to run MobileSOM, concerning scaling maps. The most common of these profiles is the Mobile Information Device Profile (MIDP). Since version MIDP2.0 a basic low level 2D graphic API is provided which for example is used to display the SOM and customised menu items on the device. Applications written for this profile are called MIDlets. A MIDlet is a Java program aimed for embedded devices that is packed inside a .jar file together with a manifest file. In the manifest minimal configurations, that have to be supported by the target device, are defined, which are the MIDP2.0 and the CLDC1.1 in case of MobileSOM. Another prerequisite to run the software is the implementation of the Multimedia Java API (MMAPI), which is an API specification for J2ME devices developed under the Java Community Process as JSR 135. Depending on how it's implemented, the APIs allow applications to play sound files like .wav or .mp3. Because of bandwidth limitations and low memory capabilities on mobile devices the prototype uses music collections in MPEG Layer-3 format. Yet there is a very little number of devices on the market available that support playing and streaming mp3's under J2ME. So far Nokia's 7710, Sony Ericsson's P900/910, M600, W950 and Benq's P50 are capable running MobileSOM. With J2ME, profiles are responsible for defining the user interface (UI) application programming interface (API). The MIDP defines two such APIs, referred to as the high-level and low-level APIs. The prototype was implemented as a low-level UI because the API gives direct access to basic graphic operations on the screen like painting circles and to input events like stylus pointing events. However, this access comes with a price, because everything shown on the screen has to be drawn manually. For example the placement and the size of menu items or fonts on devices with different resolutions need to be adjusted. A closer look to the main interfaces of the MobilSOM software is given below.

5.1 Map Interface

On the screen the SOM with a menu bar at the bottom is displayed (see left screenshot in Figure 4). The map in this example was trained with an audio collection used by Tzanetakis G. in previous experiments, presented in [10] and consists of 1000 pieces of audio equidistributed among 10 popular music genres

Fig. 4. MobileSOM on a Sony Ericsson P910 Emulator, with the Map Interace on the left and the Playlist Editor on right side

(blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock). Songs with strong beats can be found in the upper left, pieces of music with rock elements are arranged from the lower left to the middle bottom. Tracks with slower music mainly from the genres jazz and classic are placed on the lower right side. By pointing the stylus on a specific location of the map in preview mode, which is activated by default, song titles are displayed on a transparent bar located at the top. Like denoted in section 3 the finest granularity is a square unit with 10x10 pixels on the map. Mostly more than one song is mapped to a specific unit. In that case, no matter where the stylus is placed within the unit, the first song related to the unit is shown to the user. The interface was designed to be intuitive e.g. like the menu bar contains only items for applying actions that are available at the current state of the application. For example as long as there are no items in the playlist no play button is displayed. Menu items that are activated by default are the pencil, which switches between preview and playlist creation mode and the zoom-in button. There are four zoom levels for the map. As soon the zoom level increases a zoom-out button in the menu is displayed. If the map is too large for the display, navigation facilities appear in the menu bar recognizable as red arrows. Left- and right arrows are exclusively shown if it is possible to scroll horizontally, which applies to up- and down arrows for vertical scrolling as well. Clicking on the pencil item located at the bottom on the left activates the playlist creation mode and colours the pencil red. By drawing a path on the map underlying songs are added to the playlist. A path consists of particular steps, painted as filled circles. Each circle adds maximal three songs to the playlist. It is possible to release the stylus from the device and continue

drawing the path from a different location. The colour of the circles fades from white (new painted circles) to black (circle drawn at the beginning) which can be seen as a *history* function. The Path can be deleted at once by touching the rubber icon or shrinked from the top to the tail and vice versa by "mopping" the stylus on one of the list icons on the upper left menu bar. The look and feel of that function reminds of using a mop for cleaning purposes that is why this functionallity is named *the wish-mop effect* by the author. The user can switch between playlist creation and edit mode by pointing on the list icon on the very right side. Figure 5 shows all map interface items at a glance.

Fig. 5. Map interface features from left to the right: Red arrows for navigating on the map, zoom in/out glass for 4-level zooming, pencil to switch between preview and creation mode, wish-mop to cut the top or tail of the path, rubber to clean playlist at once, play/stop button for playing items from the playlist, list icon to switch to edit mode

5.2 Playlist Editor

In the Editor all songs that where selected are displayed in a list view (see right screenshot in Figure 4). If the list is to long, scrolling buttons appear in the menu bar recognizable as red arrows. For fast scrolling the stylus can be mopped on one of the arrows instead of singly pointing on them. The created playlist can be altered by removing songs or changing their position. To perform operations on an item, it must be selected first. The selected item is highlighted light green. By clicking the rubber icon the song is removed from the playlist and its successor is selected. To move a song up or down in the list, one of the list items in the upper left menu bar has to be clicked. For quickly sliding an item the stylus must mop on one of the list icons. To switch back to the map interface the map icon must be clicked.

5.3 The Player

The player is located as a transparent bar at the top of the screen and can be activated as soon as any songs are added to the playlist by clicking the play icon either on the map interface or in the playlist editor. The name of the current played song and the time progress are displayed in the player window. In editor mode the actual song is highlighted pink in the playlist. Titles can be skipped by clicking on the forward or backward menu item. The player is stopped by selecting on the stop button. There are three player modes:

Fig. 6. Editing features from left to the right: Red arrows for scrolling up and down in the list, list items for moving a selected song up or down, rubber for deleting a single item, play/stop button for playing items from the playlist, blue arrows for skipping tracks, map icon to switch to the map interface

1. play files locally
2. stream files from a server
3. play files on a remote machine

In the first case all pieces of music are available on the mobile device and can be played locally. The other modes require a network connection. An issue to overcome is that streaming of audio files is not supported by the MMAPI. The files are first downloaded to the device and played again locally. This results in long waiting times which is very obvious when the media file is big. A special feature for devices that do not implement the MMAPI is the remote mode. Music files are stored on a PC that is connected to a network and a server application is waiting for incoming item requests. After creating a playlist on the mobile device, the device acts as a remote control telling the server which song to play. It is even possible to switch between remote and local mode at runtime. An "R" placed at the right top of the play icon indicates that the song is played on another device remotely. The remote mode is switched on or off by pressing key "1" on the device. Figure 7 shows the player in the two different modes.

Fig. 7. Player in remote mode on the left and playing music from the mobile device on the right side

5.4 Emulators versus Real World Devices

The prototype was developed in Netbeans [1] an open source Java IDE that provides creating mobile applications straight out of the box. Once a MIDlet is ready to distribute, it is tested on an emulator first. Netbeans supports direct integration of emulators by various manufacturers. MobileSOM was tested on the following emulators

1. Prototype 4.0 Nokia 7710 MIDP Emulator
2. Sony Ericsson MIDP 2.0 and CLDC 1.1 P900/P910 Emulator
3. Sony Ericsson MIDP 2.0 and CLDC 1.1 M600 Emulator

Moreover the IDE supports developing applications for multiple devices by adding and executing device-specific code as configurations within a single MIDlet, called "device fragmentation". Now that the program passed all functional tests in the emulator environment there are several ways to deploy it on a device for example via Cable, Bluetooth or IR, over WAP-Push or download it directly from a server. Figure 8 shows the deployment from a MIDlet to a Nokia device. Unfortunately the deployment procedure described above turned out to be im-

Fig. 8. Deployment of a MIDlet

practicable in many aspects. The MobileSOM Player loads many Images at the beginning, which can result in an *out of runtime memory error* on real devices. A cold restart of the device is necessary. The fact that this error occurs is a sign that the software is trying to allocate more memory than is available as a single chunk. This usually occurs when a program attempts to load a large image after loading many smaller ones. The smaller images used up half the heap and the large one can't be provided a large enough chunk in which to fit. In general a program must avoid loading such large resources into memory. The only way to tell whether a MIDlet is too big to run on the phone is to test it on the actual device. Emulators cannot be accurate in simulating the heap memory management.

The Multi Media API (MMAPI) is implemented differently by every manufacturer. Most of the mobile device do not support mp3 playback at all. The most complete implementation of the JSR-135 can be found on various Sony Ericsson devices (e.g. M600 or P910) followed by some Nokia mobiles. Playing mp3's using the MMAPI works on a Nokia 7710 but it does not work on its Prototype 4.0 Nokia 7710 MIDP Emulator.

Lemma 1 (Deployment).

1. *If a MIDlet works on the emulator, it does not mean it works on the device.*
2. *If a MIDlet works on the device, it does not mean it works on the emulator.*

Finally a short evaluation of the MobileSOM running on a Nokia 7710 and a Sony Ericsson P910 is given. Because of heap memory limitations the zooming

feature is disabled. Due to the fact that streaming is performed in downloading the songs first to the device, which leads into long waiting times, all devices stored the music content locally. The audio archive contains 10 Songs from the 10 genres described above. The focus of the work relies not in evaluating the quality of the SOM but to get a feel about the interaction between a user and a mobile device with a touch screen. Clicking Icons is an easy task on the emulators but could be a tricky one on all real world devices because of calibration problems or hitting wrong icons because they are too small. That is why it has to be ensured, that the icons are big enough and well separated. Because of different screen resolutions the size of the menu items and the distance between them has to be adjusted for every device. Drawing a path is smooth on all Sony Ericsson devices but a fatigueing task on the Nokia 7710. The graphical computation power of Nokia devices is much weaker than on Sony Ericsson ones both on the emulators and the real world devices. Once a path is drawn, the wish-mop effect is tested. The feel of mopping the stylus over the touch screen is not comparable with moving the mouse over an item. In that case all real world devices show a clear advantage. The next test concentrated in the multimedia capabilities of the tested devices. Excluded is the Prototype 4.0 Nokia 7710 MIDP Emulator because of the lack of mp3 support. Songs start to play on the emulators instantly but with a noticeable delay on the real world devices. Using the mobiles as a remote control works fine on all devices. As soon they were connected to a network, remote commands worked with almost no noticeable delay.

6 Conclusions

This work presented a SOM based music player for mobile devices with new sophisticated features like mopping effects or remote controlling different devices. Because of hardware limitations the prototype cannot run in its full functionality on real world devices. Anyhow the prototype showed that the described features are practicable and will run on devices the better and the stronger their hardware becomes and the faster network connections for accessing music content are. Possible improvements are devices with more heap memory for displaying larger maps, progressive download functionality, which is a technique used to play a song while the player is still downloading and buffering the content. In near future mobile devices will also be able to perform feature extraction and the SOM algorithm on a given content without the need of external hardware.

References

1. *Using NetBeansTM IDE 5.0.* `http://www.netbeans.org/files/documents/40/718/UsingNetBeans5.0.pdf`.
2. M. Dittenbach, R. Neumayer, and A. Rauber. Playsom: An alternative approach to track selection and playlist generation in large music collections. In *Proceedings of the First International Workshop of the EU Network of Excellence DELOS on Audio-Visual Content and Information Visualization in Digi-*

tal Libraries (AVIVDiLib 2005), pages 226–235, Cortona, Italy, May 4-6 2005. http://www.ifs.tuwien.ac.at/~andi/lop.html.

3. T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995.
4. Robert Neumayer, Michael Dittenbach, and Andreas Rauber. Playsom and pocketsomplayer: Alternative interfaces to large music collections. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 618–623, London, UK, September 11-15 2005.
5. E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Master's thesis, Vienna University of Technology, December 2001. http://www.ifs.tuwien.ac.at/ifs/research/pub_pdf/pam_thesis01.pdf.
6. E. Pampalk, A. Rauber, and D. Merkl. Using smoothed data histograms for cluster visualization in self-organizing maps. In *Proceedings of the Intl Conf on Artificial Neural Networks (ICANN 2002)*, pages 871–876, Madrid, Spain, August 27-30 2002. http://www.ifs.tuwien.ac.at/ifs/research/publications.html.
7. A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, November 2002.
8. A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by musical styles. In *Proceedings of the 3rd International Symposium on Music Information Retrieval*, pages 71–80, Paris, France, October 13-17 2002. http://www.ifs.tuwien.ac.at/ifs/research/publications.html.
9. A. Rauber, E. Pampalk, and D. Merkl. The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research*, 32(2):193–210, June 2003. http://www.extenza-eps.com/extenza/loadHTML?objectIDValue=16745\&type=abstract.
10. G. Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Princeton University, 2002.