Music Genie: Interactive, Content-Based Browsing of Music Based on Thumbnail Playback

Ramin Samadani, Tong Zhang
Media Technologies Laboratory
HP Laboratories Palo Alto
HPL-2007-38
March 12, 2007*

music discovery, thumbnails, interactive browsing

Music Genie refers to audio processing technologies for quick and easy music browsing and discovery. We built research prototype software that allows users to quickly generate playlists from a collection of thousands of songs. Our software provides an interactive, content-adaptive, thumbnail-based presentation of music. The software accepts user inputs interactively during use, and it consecutively presents short music thumbnails for quick browsing. This direct, interactive presentation of music makes our system simple and transparent for users. Unobstrusively, the user's interactions (or lack thereof) are input into a real time system that, based on the prior analysis of the music content, adaptively presents a potentially interesting new song thumbnail. This technology may be used to 1) assist users discover music for purchase from a web music store; 2) allow users to browse their PC music collection based on current taste; and 3) allow intelligent browsing, playback or playlist generation in small, interface-limited, music players.

* Internal Accession Date Only
Approved for External Publication

Music Genie: Interactive, Content-Based Browsing of Music Based on Thumbnail Playback

Ramin Samadani and Tong Zhang

Abstract

Music Genie refers to audio processing technologies for quick and easy music browsing and discovery. We built research prototype software that allows users to quickly generate playlists from a collection of thousands of songs. Our software provides an interactive, content-adaptive, thumbnail-based presentation of music. The software accepts user inputs interactively during use, and it consecutively presents short music thumbnails for quick browsing. This direct, interactive presentation of music makes our system simple and transparent for users. Unobstrusively, the user's interactions (or lack thereof) are input into a real time system that, based on the prior analysis of the music content, adaptively presents a potentially interesting new song thumbnail. This technology may be used to 1) assist users discover music for purchase from a web music store; 2) allow users to browse their PC music collection based on current taste; and 3) allow intelligent browsing, playback or playlist generation in small, interface-limited, music players.

1 Introduction

More and more people enjoy music from high-capacity portable players and from internet music services. Developing natural methods for exploring the growing music collections is now a critical need. Two trends make music exploration increasingly relevant: 1) individuals' digital music collections are increasingly prevalent and large (hundreds to tens of thousands of songs); and 2) digital music production tools have made music creation simple, with many new, unknown artists producing exciting new music that grows the universe of available music. Music exploring difficulties commonly occur in two settings: the discovery of desired songs from an unkown collection, and the browsing of one's own large music collection.

This report describes our approach to music discovery and browsing by the direct audition of music thumbnails. We believe that directly presenting music during music discovery is natural for users. Section 2 discusses current approaches to music discovery. Section 3 discusses the quick playlist generation demonstration software developed using Music Genie subsystems. Section 4 discusses the offline processing and online processing subsystems used to create the demonstration of Section 3.

2 Current approaches

A current approach to music exploration, such as the example found at **http://www.amazon.com**, uses web-based interfaces to allow text-based search and exploration of music, together with the option of playing samples after the search is completed. This approach is mostly text-based, and the audition of song clips is only used as a verification step. We believe the text-based approach may be cumbersome, and that users may prefer, after a first preliminary text-based preselection by artist or category, to directly hear many potential candidate songs.

Another technology currently used for music exploration, with an example also found at **amazon.com**, is collaborative filtering, which analyzes commonality in prior music selections of users, to derive new music recommendations. Collaborative filtering is known to have difficulties recommending lesser-known artists until there is sufficient history of purchase of the artists' songs. Currently, the granularity of the recommendations by collaborative filtering may be coarse: by artist. The increasing trend is for users to buy songs individually, making collaborative filtering more difficult to use in this new setting. In addition, the ease with which artists create new music means that the initiation problems of collaborative filtering will become more severe with increasingly new music.

A third current approach, used by Pandora, involves manual annotations by trained experts. Similarities between the different manual annotations are used to browse through music. The manual process is time-consuming and may become a bottleneck as the growth of new music continues to accelerate. In addition, keeping consistent annotations when different people do the annotation complicates the annotation process.

3 Playlist Generation Example

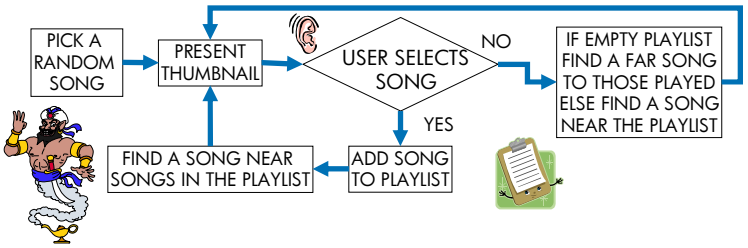

Figure 1: The operation of the playlist generator

Music Genie analyzes music content automatically. Using automatic analysis allows the system to scale with growing music collections. It also allows the inclusion of lesser-known artists during discovery. For example, Music Genie may even recommend a song not yet heard by any user.

An example application is described in this section: a computer-assisted playlist generator. We believe users find it natural to directly select music while listening to song candidates. For this listening process to be quick, we present short thumbnails that are representative of the songs. Simple user inputs during the selection adapt the system to the user's current interest.

Figure 2: The user interface for the playlist generator shows the currently playing song as well as several songs that have been added to the playlist

Figure 1 shows the program logic for our playlist generator. When the program starts, a random song from the unknown collection is selected and its thumbnail is played. The default length of the thumbnail playback is eight seconds. If the user takes no action in those eight seconds, or actively decides against the song by selecting *no*, the system starts to play a song thumbnail far in feature space to the thumbnails played previously. This mode of selecting far songs from the ones heard continues, allowing the user to quickly explore the variety of the music collection, until the user selects *yes* for a song. At this point, this first song is added to the playlist, and the mode

of operation changes to find songs near to any of the songs selected so far. Songs for which the user selects *yes* are added to the playlist.

The system shown in Figure 1 adapts in real-time to the user's inputs. During operation, the next song presented is selected based on the prior *yes/no* user choices. Thus, songs that the user selects at a particular time depends on current user inputs, and may not be the same as those selected at another time.

Figure 2 shows the user interface for the playlist generator. Once the application starts, the currently playing thumbnail is displayed in the upper display window with the label *Song currently playing*. The buttons labeled *I Like It* and *Skip* correspond to the *Yes* and *No* selections, respectively, of Figure 1. The bottom panel of Figure 2 shows the playlist songs selected so far. In the figure, five songs have been selected, and their paths are shown. Once playlist selection is complete, users may select a song to play in full, or select *suffle play* to play a random one of the selected songs. In a commercial application, once the playlist is complete, the songs could be purchased.

In summary, our approach is based on the music content. The user interacts directly with music thumbnails allowing a quick, direct exploration of the music collection. Depending on the implementation, our system may be used to explore a collection of up to several thousand songs. Alternately, our system may be used for discovery from a larger collection by first pruning the larger collection using existing text-based search, and then using our system to explore this preselected song collection based on direct thumbnail-based music playback.

4 Offline and online subsystems

The playlist generator of Figure 1 is enabled by the *offline* and *online* audio processing subsystems now described. The offline processing occurs prior to system use. The online processing occurs during system use, and as tested on PCs, meets a real-time interactive requirement. The subsystems described in this section may be used in different ways, and they are also suitable for applications other than the example described in the previous section.

4.1 Offline subsystem

The offline processing applies to unencrypted MP3 files. The MP3 files are converted to WAV files before being processed by our offline components. Currently, the processing applies in batch to a whole directory tree, but it should be straightforward to apply the processing to each song as it is added to the database. For each song, the following operations are performed:

1. Extract song path information.
2. Extract ID3 tags. This information may be useful for displaying.
3. Extract thumbnail starting point for each song.
4. Extract music features.

We use structural analysis of the song to extract a representative thumbnail. The music features are extracted by computations on the spectrogram, capturing characteristics roughly associated with beat strength, pitch, spectral timbre and tempo. The thumbnail extraction and the feature extraction software is written in C.

The amount of information that needs to be stored for each song is small. The current implementation stores all the offline processing information in a text file. For one example, with 1388 songs on a laptop, the text filesize was around 360 kbytes, and most of the size was taken by the redundant representation of song paths.

4.2 Online subsystem

The playlist generator of Figure 1 used some of the components of the online subsystem shown in Figure 3. The online subsystem accepts user inputs and provides audio and display outputs, as well as controls the algorithm that select next songs given user inputs for the current session. The online system is written in Perl, with multiple processes enabling experimentation with real-time interaction.

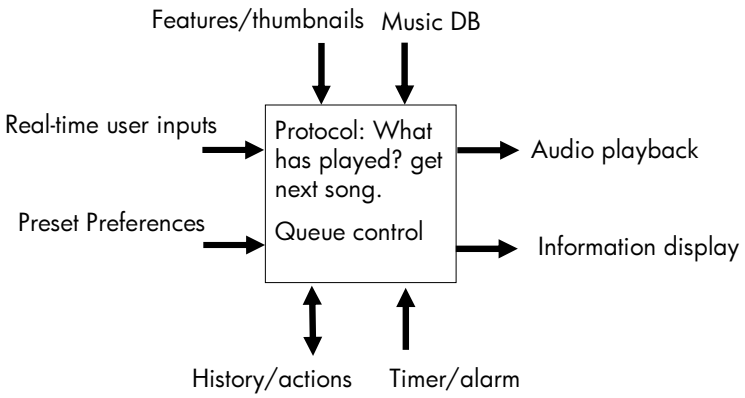

Figure 3: The information flow for the online subsystem

First we discuss the input/outpuf flows of Figure 3. The arrow labeled *Real-time user inputs* accepts user inputs. In different implementations, this information has come either from keyboard inputs or from socket inputs. The arrow labeled *Audio Playback* accesses the music, labeled *Music DB*, as well as the thumbnail starting point, labeled *Features/thumbnail*, and plays the thumbnails. The *Timer/alarm* controls the length of the thumbnail playback by interrupting playback after a set time. The *Information Display* was used in some of the demonstrations to display song ID3 information to the user, but it is not used for the example in Section 3. The *History/actions* recorded user actions and system actions for other experiments, but it is not used in the application of Section 3.

The online subsystem also calls the *next-song algorithm* that decides what song to present next. This algorithm processes the audio features, accessible through the arrow labeled *Features/thumbnail*. It also takes as input the information found at the arrow labeled *Preset preferences*. For example, with the present preferences, one may control the distance of near and far songs, as well as turn on or off different audio features for experimentation and tuning. The next-song algorithm is implemented in C, and it is called by the Perl controller of the online system. The algorithm is based on the user feedback, and the song features extracted by the offline system described in Section 4.1

The *Preset preferences* also control other aspects of the online system, including the path of the features file and the length of the thumbnails during playback.

5 Conclusions

Experience with the software prototypes leads us to believe that the pattern recognition and feature extraction techniques, combined with the direct audition of short thumbnails, provides users with natural, pleasing music discovery or browsing experiences. The technology could be used for music sales from web sites, music player software, and music player devices.