Exploring Music Collections on Mobile Devices.

Olga Goussevskaia
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
golga@tik.ee.ethz.ch

Michael Kuhn
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
kuhnmi@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

Ever larger collections of music are stored on mobile devices. The process of managing these repositories therefore becomes increasingly challenging. In this work we propose to use a map of the "world of music" as a data structure for music exploration and retrieval on mobile devices. We present *Mobile Music Explorer*—a mobile application, which allows users to create playlists by specifying trajectories on the map and to use similarity based search methods to navigate through their personal music collections. Our navigation methods ensure that any part of the collection can quickly be reached, even for a large set of items. Moreover, we show that the map representation is a natural approach to provide efficient and distributed operation.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: General

General Terms

Design, Performance.

1. INTRODUCTION

Mobile phones have been turning into multi-purpose entertaining devices with increasing storage capacity and ever better audio codecs for high-quality music reproduction. Ever larger collections of music are stored on mobile devices, making the process of managing these repositories more challenging. The traditional browsing of folder hierarchies and search by song title or album tends to be insufficient to maintain an overview of a collection of orders of thousands of tracks. Search methods based on song *similarity* offer an alternative to keyword-based searches, allowing users to abstract from manually assigned metadata, such as album title or, frequently imprecise or incorrect, genre information. Moreover similarity-based organization allows personal collections to be seen not just as isolated lists, but positioned in the global context of the "world of music", i.e., it becomes possible to relate personal music collections to larger collections, containing tracks listened to by other users and newly-released music.

The demand for novel music management tools has been captured by many commercial projects. Online music communities, such as *Last.fm*, *iLike*, and *Pandora* have been offering novel ways to discover and experience music. By managing millions of user profiles, they apply collaborative filtering techniques to search for music, generate playlists, and recommend similar music. Most of these technologies are based on large centralized databases and thus not suited for mobile environments.

In [3] we propose to use a map of the "world of music" as a data structure for music exploration and retrieval. Based on collaborative filtering techniques, we constructed a map, where tracks are assigned to points in a Euclidean space, and distances between them represent music similarity. We showed that similar music is grouped together and there exists a correspondence between regions on the map and specific genres, such as rock, jazz, or hip hop. In this work we show that such a map representation is a perfect candidate for organization of large collections of items and efficient distributed operation. On the one hand, it serves as a basis for intuitive navigation, such that reachability and searchability requirements are met. On the other hand, it allows distributed computation of music similarity. In order to compute the similarity between two songs it is enough to calculate the distance between the coordinates assigned to them. No memory-expensive data structures need to be maintained on the mobile device and no access to a centralized server is required. Therefore, this approach also fits naturally for distributed file sharing and mobile social interaction based on users' music profiles.

We developed *Mobile Music Explorer*—a mobile application that allows users to navigate through their music collections and create playlists that span trajectories on the map of music. The rest of this paper is organized as follows. In Section 2 we discuss some related work, in Section 3 we present a brief overview of the map data structure, in Section 4 we describe the architecture and the main functionalities of *Mobile Music Explorer*, and finally in Section 5 we discuss the future directions for our work.

2. RELATED WORK

The problem of defining music similarity has been addressed from three main perspectives: analysis of manually added meta-data [1, 9, 11], analysis of acoustic properties extracted from the audio signal [5, 7, 8, 10], and collaborative filtering [2, 12]. Methods that rely on meta-data

MobileHCI 2008, September 2-5, 2008, Amsterdam, the Netherlands
Copyright 2008 ACM 978-1-59593-952-4/08/09 ...$5.00.

or audio signal of each track usually suffer from scalability problems, given that the data is difficult and expensive to obtain. Methods based on collaborative filtering are intrinsically more scalable, given that no analysis of individual audio files is required. Despite being based on collaborative filtering like our work, the work in [2] focused on similarity of artists, which cannot be directly applied to managing music collections. In [12] the similarity computation is based on shortest path algorithms in large graphs, which is not suitable for hardware-constrained devices.

Visualization of music collections using self-organizing maps was addressed in [5, 8]. Both approaches differ in two ways from our work. First, they rely on audio feature extraction to define similarity. Second, they miss the advantages of high-dimensional spaces in terms of accuracy.

Playlist generation has been addressed in a variety of ways. In [10], playlists were generated using traveling salesman algorithms; in [12] playlists were generated by exploring the graph neighborhood of a given seed-song; in [9] playlists were constructed based on users skipping behavior. *Audioscrobbler* uses collaborative filtering to recommend playlists according to users with similar listening behaviors. The idea of using trajectories on a map was explored in [8] to construct playlists with smooth transition in acoustic characteristics. Our method differs from these approaches either because it scales to a much larger universe of tracks, or because it allows distributed operation on mobile devices.

3. THE MAP OF MUSIC

In [3] we developed the *Music Explorer* web application[1], which uses an automatic method to derive music similarity and scales to millions of titles, not requiring the analysis of audio files or manually-generated metadata. The approach is based on user listening patterns, extracted from a publicly available source. Similarly to how Amazon uses the fact that two items are related because they have been purchased by the same person [6], we assume that two songs are related if they are frequently listened to by the same user.

Pairwise co-occurrence values cannot be used directly, since maintaining a data structure of size $\Theta(n^2)$, especially on a mobile device, is prohibitive. Similarly to other existent solutions, we created an intermediate sparse graph, in which similarity between two songs is defined as the weight of the shortest path between them. However, since in order to efficiently calculate shortest-paths, the whole graph has to be in main memory, operating directly on it remains prohibitive for a mobile device.

We thus went one step further and embedded the graph into a 10-dimensional (10D) Euclidean space. An embedding is a mapping of vertices of the graph into points in the Euclidean space, such that all pairwise graph distances are approximately preserved. In the course of our work we observed that less than 10 dimensions is insufficient for a collection of this size, since it results in bad quality similarity values. Having the coordinates of each node, the graph structure is not needed anymore.

The map representation not only allows fast and memory-efficient computation of music similarity (both time and memory complexities to calculate the distance between two items is $O(1)$, as opposed to $O(m)$ memory and $O(n \log n)$ time to compute a shortest path in a graph), but also enables new functionalities, which we discuss next.

[1]`www.musicexplorer.org`

4. MOBILE MUSIC EXPLORER

Mobile Music Explorer is comprised of three main components: the global map of the "world of music", the local music collection of the user, and the application software running on the mobile device. The coordinates correspondent to the map of the "world of music" were made available through a web service at `www.musicexplorer.org`. Whenever the music collection on the mobile device changes, a database update is performed. Thereafter all the operations can be performed offline.

Mobile Music Explorer was implemented using *Android*, a mobile phone platform developed by the *Open Handset Alliance*, a business alliance comprising Google, Intel, Motorola, and other companies in the mobile handset market, whose goal is to develop open standards for mobile devices.

In the rest of this section we describe two main functionalities already implemented on *Android*: *playlist generation* and *navigation*. We conclude the section by sketching a third functionality, which is still in the process of porting from a desktop version: *visualization in 2D*.

4.1 Playlist Generation

The local map of music (in 10D) allows the user to generate playlists by exploring the neighboring area of a track or by following a trajectory between two (or more) tracks on the map. In fact track titles do not need to be used as anchor points. If a graphical visualization is available, areas of interest can be selected graphically, as points or regions on the map. The idea is that by spanning the space between different regions on the map, the playlists will reflect gradual transitions between genres associated to these regions.

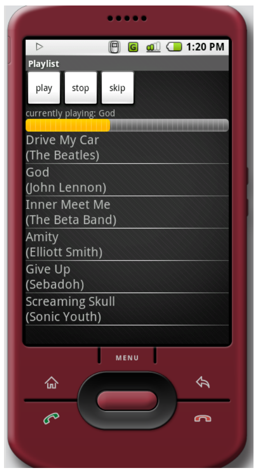
The user interface was implemented as follows: first, the user enters the titles of the start and the end tracks of the playlist, then the user can specify the duration (in minutes) or the number of tracks, and allow or disallow duplicate artists. Playlists are generated by a simple greedy algorithm. For each segment, the trajectory is divided uniformly according to the requested number of songs. Then to each of the resulting points the closest song not contradicting any constraints is selected. If the length of a playlist is defined by duration rather than the number of songs, the requested number of songs is derived from the average song length in the subset. An example playlist is shown in Figure 1(a). It can be seen that the playlist presents a genre transition between the start point (The Beatles' "Drive my car") and the end point (Sonic Youth's "Screaming Skull").

4.2 Navigation through Music Collections

The Euclidean representation offers the advantages of similarity-based search. As opposed to keyword-based search, it allows to retrieve new items, whose titles the user might not know, based on their similarity to known items, from which the user starts to navigate. Such "proximity-based" search is particularly important in a context, where music collections become larger and more dynamic.

The problem of list-based navigation can be described as follows. Given a collection of n items (with their coordinates), a device screen on which a list of k ($k \ll n$) items can be displayed, as in e.g. Figure 1(a) (we refer to this list as *displayed-list*). How can we efficiently navigate through all n items? Two basic requirements have to be met:

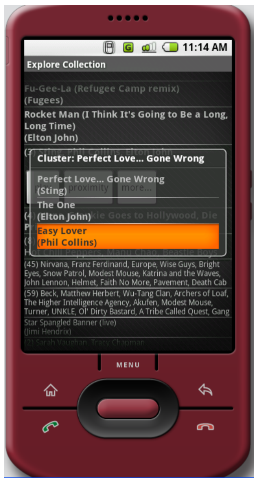
- *Reachability*: The entire collection should be reachable from any given starting point.



currently playing: God

Drive My Car
(The Beatles)
God
(John Lennon)
Inner Meet Me
(The Beta Band)
Amity
(Elliott Smith)
Give Up
(Sebadoh)
Screaming Skull
(Sonic Youth)

(a) A playlist.



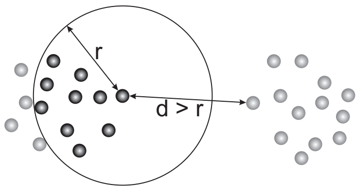
Fu-Gee-La (Refugee Camp remix)
(Fugees)
Rocket Man (I Think It's Going to Be a Long, Long Time)
(Elton John)
(3) Sting, Phil Collins, Elton John

Cluster: Perfect Love... Gone Wrong
Perfect Love... Gone Wrong
(Sting)
The One
(Elton John)
Easy Lover
(Phil Collins)

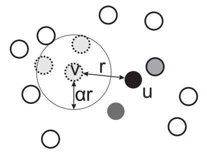
(45) Nirvana, Franz Ferdinand, Europe, Wise Guys, Bright Eyes, Snow Patrol, Modest Mouse, Katrina and the Waves, John Lennon, Helmet, Faith No More, Pavement, Death Cab
(59) Beck, Matthew Herbert, Wu-Tang Clan, Archers of Loaf, The Higher Intelligence Agency, Akufen, Modest Mouse, Turner, UNKLE, Ol' Dirty Bastard, A Tribe Called Quest, Gang
Star Spangled Banner (live)
(Jimi Hendrix)
(2) Sarah Vaughan, Tracy Chapman

(b) Clicking on a group.

Figure 1: Screenshots of *Mobile Music Explorer*



(a) Getting stuck in a cluster.



(b) Song grouping.

Figure 2: List-based navigation.

- *Searchability*: Users should be able to quickly find what they are looking for. In particular, short paths should exist between any two songs. Moreover, the users should be able to detect these short paths, i.e. at any stage they should be able to select an item that brings them considerably closer to the goal.

An intuitive approach to explore a collection would be to select a starting point (or *root*) and to present the proximity of this point in the *displayed-list*. Clicking on an item in the *displayed-list* allows to change the root and display this new root's proximity.

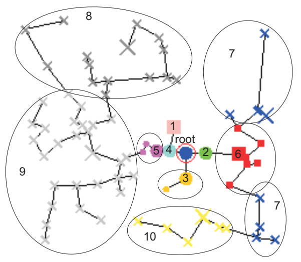
A naive solution might always put the k closest neighbors of the root song in the *displayed-list*. Unfortunately, this approach has several drawbacks. First, it might take a considerable number of clicks to move from one end of the map to the other. Second, if the tracks are not uniformly distributed in space, but form clusters, there is a risk of getting stuck in a cluster. This problem arises, if the diameter of a cluster consisting of more that k tracks is smaller than the distance to the closest track outside, as illustrated in Figure 2(a).

We will next discuss two solutions to the problem of list-based navigation: *neighborhood clustering* and *small-world navigation.*

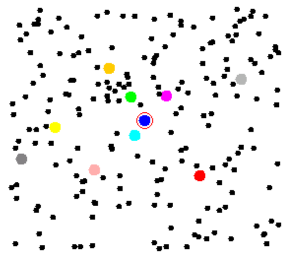
4.2.1 Neighborhood Clustering

Neighborhood clustering addresses the issues of reachability and searchability in three stages:

1. Forming local groups: In order to allow large distances to be bridged with each click, each item in the *displayed-list* is associated with a group of tracks, instead of only one



(a) Neighborhood clustering: 3 stages (1:circles, 2:boxes, 3:crosses). Each number denotes one item in the *displayed-list.*



(b) Small-world links: Nodes close to the root are more probable to be included in the *displayed-list* than nodes far away.

Figure 3: List-based navigation.

track. Assume that a *displayed-list*, starting at a root track u, contains a neighbor v, s.t. $d(u,v) = r$. Then all the tracks inside the ball with radius αr $(0 < \alpha < 1)$ around v are grouped together in a *local* group g_i, and displayed as one single item. A group g_i is therefore comprised of a representative song, v in this case, and its neighbors $w, d(v,w) < \alpha r$. Each group g_i is formed by successively selecting the representative song v_i as the closest node around the root u that has not yet been assigned to any group $g_j, j < i$. The basic idea is illustrated in Figure 2(b) (u is the root, and the three light gray nodes belong to the group of v).

2. Building a Minimum Spanning Tree (MST): In order to guarantee local connectivity, an MST is constructed over the n points in 10D representing the user's collection. This is done only once, right after acquiring the coordinates. Let $g(u)$ be the set of all songs assigned to a group item g_i of a *displayed-list* with root u. Moreover, let R denote the maximum distance $d(u,w)$, $w \in g(u)$. Then, any nodes adjacent to an edge (in the MST) crossing the ball with radius βR $(0 < \beta < 1)$ around u are added to the *displayed-list* as new group centers. These new local groups are formed the same way as in (1). Observe that all nodes adjacent to u in the MST, even those located outside the ball of radius R, are included in the *displayed-list* of u. Since the MST connects the entire point set, any track becomes reachable.

3. Clustering distant nodes: In order to guarantee long distance searchability, a few additional *long-distance* groups, that cover all songs not yet assigned to a local group g_i, are inserted in the *displayed-list.* For each local group g_i, formed in steps (1) and (2), a centroid c_i is placed at the position of the representative song v_i. Then the *k-means* algorithm is applied to assign all remaining long-distance songs to one of the centroids. The representative song of such a newly created group becomes the song closest to the group's center of mass. Observe that after this step, all n songs are assigned to one of the k group-items in the *displayed-list.* To make the long distance groups manageable, they are labeled by a list of most representative artists. Moreover, to visually distinguish long-distance groups from local groups, they are displayed in smaller font and in the bottom of the *displayed-list.* Long-distance groups are used for large distance jumps (i.e. to leave the current area on the map and explore another one). The effect of the entire 3-stage process is illustrated in Figure 3(a). Moreover, Figure 1(b) shows a screenshot of a sample *displayed-list* on the mobile device.

4.2.2 *Small-World Navigation*

Stanley Milgram's "Six-Degree-of-Separation" experiment did not only reveal that people are interlinked by astonishingly short acquaintance chains, but also that these short paths can efficiently be discovered by humans disposing of local knowledge only. Jon Kleinberg has later recognized that a specific edge length distribution ($1/r^d$, where d is the dimensionality of the underlying space) is required to achieve this navigability properties [4]. He has shown that augmenting a grid (or the higher dimensional equivalent) by a single random outgoing link per node is enough to ensure poly-logarithmic path lengths between any pair of nodes. Moreover, he has shown that these short paths can be detected using local knowledge only.

We take advantage of these insights by artificially overlaying our data with such an edge length distribution. Whenever a user selects a track, k random tracks are selected according to Kleinberg's distance distribution. These tracks are then sorted according to their distance from the root song, and the resulting list is presented to the user. Due to the specific distance distribution, the first items in the list are close to the root track, whereas the last items build the entry points to discover new areas farther away. Figure 3(b) illustrates the distribution of the listed tracks in a 2D sample space. The *displayed-list* is newly calculated every time the user selects a root song. As a consequence, every track has a certain probability to appear, which ensures that any song can eventually be reached. Moreover, if a user is not happy with the "neighbors" offered in the *displayed-list*, he can simply re-select the same root and hope for a better list.

While this method might fail to quickly discover a particular song (as it might get missed by the random process several times), it provides an elegant light-weight solution to quickly get an overview of a music collection.

4.3 Visualization in 2D

Although the embedding of the music graph was done in 10D, our experiments showed that a reasonable visualization in 2D can be achieved by working with the first two dimensions. Of course the quality of the embedding deteriorates, and some points that were originally far away in 10D become closer or even overlap when projected into a 2D plane.

Besides the dimensionality reduction, another issue that has to be addressed when visualizing a large collection of songs is the limitation of the device's screen. We implemented a hierarchical visualization of the 2D map, by fixing the number of songs displayed in each square unit and giving the user the ability to zoom in and out of the map. In a certain area, from top to bottom level of zooming, tracks are displayed in order of decreasing popularity (or playcount). As a result, the entire music collection can be visualized as points in space on a relatively small display.

We implemented this functionality as a desktop version, but we are in the process of porting it to the mobile platform. Additional issues, such as even smaller screen size, different input devices (e.g., no mouse for zooming), and performance, have to be considered.

5. FUTURE DIRECTIONS

We believe that the map of music has the potential to trigger a number of novel applications, improving the way music is currently treated. In this section we outline some of such application scenarios.

The region on the map occupied by a personal collection can be compactly encoded as a volume (or a union of several volumes). This could facilitate an autonomous file sharing application: Whenever two mobile devices come into connection range (by means of e.g. Bluetooth), they exchange the volumes representing their local collections and share whatever songs are in the volume intersection but not available on both devices. The only information the devices require for such applications are the coordinates of each song.

Coordinates might also give way for innovations in the entertainment and event industry. Imagine more comprehensive systems that directly and autonomously interact with the audience. During a party, for example, the system could collect the volumes of people's favorite music. Sophisticated devices could provide even more feedback. Recent mobile phones, for example, are equipped with motion sensors, that could measure the fraction of dancers in a given moment. Such information could then be used to find the optimal mixture of music for a given audience.

6. ACKNOWLEDGMENTS

We would like to thank Anitha Gonukula, Philipp Kuederli and Samuel Pasquier for their contributions in the development of 2D visualization module.

7. REFERENCES

[1] J. Aucouturier and F. Pachet. Scaling up Music Playlist Generation. In *ICME*, 2002.

[2] M. R. David Gleich, Leonid Zhukov and K. Lang. The World of Music: SDP layout of high dimensional data. In *InfoVis*, 2005.

[3] O. Goussevskaia, M. Kuhn, R. Wattenhofer, and M. Lorenzi. From Web to Map: Exploring the World of Music (Under submission).

[4] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, pages 163–170, New York, NY, USA, 2000. ACM Press.

[5] P. Knees, M. Schedl, T. Pohle, and G. Widmer. An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *ACM Multimedia*, pages 17–24, 2006.

[6] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[7] B. Logan. Content-based playlist generation: Exploratory experiments. In *ISMIR*, 2002.

[8] R. Neumayer, M. Dittenbach, and A. Rauber. PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *ISMIR*, pages 618–623, 2005.

[9] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, pages 634–637, 2005.

[10] E. Pampalk, T. Pohle, and G. Widmer. Generating similarity-based playlists using traveling salesman algorithms. In *DAFx*, 2005.

[11] J. Platt. Fast embedding of sparse music similarity graphs. In *NIPS*, volume 16, 2004.

[12] R. Ragno, C. J. C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *MIR*, pages 73–80, 2005.