# From Web to Map: Exploring the World of Music

Olga Goussevskaia
ETH Zurich, Switzerland
golga@tik.ee.ethz.ch

Michael Kuhn
ETH Zurich, Switzerland
kuhnmi@tik.ee.ethz.ch

Michael Lorenzi
ETH Zurich, Switzerland
mlorenzi@student.ethz.ch

Roger Wattenhofer
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

## Abstract

*Ever growing music collections ask for novel ways of organization. The traditional browsing of folder hierarchies or search by title and album tends to be insufficient to maintain an overview of a collection of orders of thousands of tracks. Methods based on song similarity offer an alternative to keyword-based search. In this work we propose to use a high-dimensional map of the "world of music" as a data structure for music retrieval and exploration of personal collections. Our approach does not require expensive analysis of audio signals and scales to hundreds of thousands of tracks. The techniques presented in this work can be used in a variety of applications, ranging from automatic DJs to file sharing on mobile devices. As a concrete example, we have developed a web-application that allows users to visualize and navigate through their music collections and create playlists by specifying trajectories.*

## 1. Introduction

It is widely accepted that media usage is changing rapidly these days. This process has been ignited by several technological advances, in particular, the availability of broadband internet, the world wide web, affordable mass storage, and high-quality media formats, such as `mp3`. All this enabled the digital music revolution about 10 years ago.

Many music lovers have now accumulated collections of music that have reached sizes that make it hard to maintain an overview of the data by just browsing hierarchies of folders and searching by song title or album. Search methods based on song *similarity* offer an alternative, allowing users to abstract from manually assigned metadata, such as, frequently imprecise or incorrect, genre information. In a context where music collections grow and change rapidly, the similarity-based organization has also the advantage of providing easy navigation and retrieval of new items, even without knowing the new songs by name. Moreover, it allows personal collections to be seen not just as isolated lists, but positioned in the global context of the "world of music", i.e., to relate personal music collections to larger collections and newly-released music. This opens possibilities, such as sophisticated recommendations, context-aware retrieval, and discovery of new genres and tendencies.[1]

This work introduces the concept of a Euclidean map as a basic data structure for music exploration and retrieval. In contrast to existing approaches that go into similar directions, the primary goal of our map is not visualization. Rather, it aims at adequately reflecting, and compactly representing similarity between songs and genres. Besides providing intuitive examples, such as pictures of the map, neighborhood lists of songs, and sample playlists, we also provide a quantitative analysis of how well this map captures music similarity. We show, for example, that neither two nor three, but rather ten or more dimensions are required to appropriately map the world of music.

We describe a 3-step process to derive such a map from the information contained in the music community site *last.fm*: First, we estimate song-to-song similarities using collaborative filtering techniques. Second, we construct a graph (the *web of music*) from these pairwise similarity values. Third, we map the graph to Euclidean space while approximately preserving distances. For the last step, we introduce a novel technique (called *iterative embedding*), which improves on existing algorithms. The result is a map comprised of more than 400K songs—an application foundation, which to the extent of our knowledge is more comprehensive than other existing approaches.

We will sketch a multitude of possible application scenarios of such a map, covering music retrieval, explo-

[1] All this holds for media in general, not only for music. However, we believe that music will again be at the forefront of development, as visual media often experiences additional technological challenges.

ration and organization. To make things more concrete, we have implemented a web application to illustrate some of these functionalities. A working prototype is available at `www.musicexplorer.org`.

## 2. Related Work

Music similarity has been addressed in numerous studies, and only a small subset of them is mentioned here. There are three main strategies to obtain similarity information: audio content analysis [11, 15, 18, 19, 20], metadata analysis [2, 21, 22, 23], and collaborative filtering [4, 24].

Manually created metadata and audio content features are usually difficult and expensive to obtain. Although there are projects aiming at constructing publicly available databases with detailed metadata of large amounts of music titles, e.g. `freedb.org`, most of the existent approaches present experiments on datasets of size ranging between 2K and 20K titles [2, 15, 19, 20, 21, 23]. An example of experiments on a bigger dataset is presented in [22].

Similarity measures based on collaborative filtering typically explore publicly available usage information and, therefore, are more scalable. In [24] a similarity measure based on co-occurrence of songs in professional radio streams is proposed. The final similarity values are then directly queried from a graph with $60.5K$ nodes, which, in contrast to our map-approach, is not suitable for hardware-constrained devices. Another example of applying user-based similarity information can be found in [4].

The evaluation of music similarity methods can be roughly classified in acoustic-based [1, 16, 19] and "subjective" [3, 6, 21, 22]. Acoustic analysis is an objective measure, however, it does not necessarily reflect the perception of the listeners. A comprehensive comparison of acoustic and subjective measures is presented in [3], which indicates that deriving music similarity from co-occurrence in personal music collections is the most reliable approach.

The visualization of music collections has been addressed in [11, 18]. Both approaches use self-organizing maps and define similarity based on audio feature extraction. The work in [11] focused on creating a virtual reality experience, as opposed to managing music repositories. In [18], an interactive map to organize music collections on mobile devices is presented. It differs from our approach in that their map is audio-feature-based (as opposed to social-oriented), and, similarly as [11], misses the advantages of high-dimensional spaces in terms of accuracy.

Playlist generation has been addressed in a variety of ways, such as by using traveling salesman algorithms [21], by exploring graph neighborhoods [24], or by analysing skipping behavior [20]. The idea of using trajectories on a map to construct playlists with smooth transition was explored in [18] and [22]. *Audioscrobbler*, finally, uses collaborative filtering to recommend playlists. Our method differs from the mentioned approaches in that it scales to a larger universe of tracks, allows distributed operation on hardware-constrained devices, and/or takes advantage of high dimensional space to improve the quality of the similarity measure. In [8] we developed a proof-of-concept mobile application that incorporates these features and demonstrates their usefulness in the outlined application scenarios.

## 3. From Perception to Web

To obtain similarity values between songs we rely on collaborative filtering techniques. The fact that two items are related because they frequently co-appear in usage data has shown to work well in previous studies [12, 13, 14]. Similar to how Amazon uses the fact that two items are related because they have been purchased by the same person, we assume that two songs are related if they are frequently listened to by the same user. We have gathered the required usage information from *last.fm*[2], which is a music-community site that counts over $20$ million users, and records each user's listening patterns. In particular, for each user, the $50$ most frequently listened songs can be queried. We will refer to these lists as *top-50 lists*. We have crawled more than 290K of these lists that contain a total of more than $1.5$ million distinct songs. We assume that songs that co-appear in such a list exhibit some degree of relatedness, in the same way as items that are bought by the same person do. Observe that exceptions to this rule are typically random. I.e., it is about equally likely that a person listens to Bach and U2, as it is that a person listens to Bach and Eminem. In the co-occurrence analysis, such "errors" therefore result in random noise, which does not significantly affect the outcome.

Observe that simply counting the number of co-occurrences of two songs to calculate the pairwise similarities overestimates the similarity of popular songs, as they clearly have a higher probability of appearing in the same top-50 list (due to their high number of individual occurrences). To overcome this problem, some sort of normalization is required. Several coefficients have been proposed that address this issue [17]. We have compared the performance of *cosine*, *dice*, *jaccard* and *overlap* coefficients, and found that the cosine coefficient performs best in our setting. The cosine coefficient is defined as $n_{i,j}/\sqrt{n_i n_j}$, where $n_i$ denotes the number of occurrences of song $i$, and $n_{i,j}$ is the number of co-occurrences of songs $i$ and $j$.

Applying the inverse of the cosine measure $(1/c)$ to all pairs of songs results in a graph that contains an edge between any two songs that appear together in at least one top-50 list. To get rid of random effects any edges origi-

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

nating from a co-occurrence value of less than 2 have been removed. This step also eliminates any songs that occurred only once. Even after this step, the graph is extremely big, making it difficult to handle. Therefore, it has been sparsened using an edge weight threshold, which was defined such that the overall connectivity (i.e. the size of the largest connected component) was only marginally affected. The result is a graph $G$ which contains $n = 430,000$ nodes and $m = 6,300,000$ edges. Using this graph, the similarity between songs is approximately given by the shortest path between them.

## 4. From Web to Map

Due to the large size of our "web of music" even simple operations, such as shortest-path calculations, are computationally expensive. In order to efficiently use such a large graph in (possibly mobile or distributed) applications, we go one step further and create the "map of music", which is an *embedding* of the graph into a Euclidean space. An *embedding* is the assignment of coordinates to each node of the graph. In our case, the goal is to approximately preserve all pairwise distances. That is, an assignment of coordinates is sought, such that the ratio $d_G(i,j)/d_E(i,j)$ between the graph distance $d_G$ and the embedding distance $d_E$ is approximately one for all node pairs $(i,j) \in G$.

To compute the similarity between two songs based on a graph demands for a costly shortest path calculation if the songs do not happen to be neighbors.[3] This shortest path evaluation does not only imply long calculation times but also exhibits an extremely high memory footprint. Having an embedding, the (Euclidean) distance between songs can directly be computed from their coordinates, i.e. in $O(1)$ time and with $O(1)$ memory consumption. No information about any other songs or structures is required. Embeddings are thus particularly well suited for distributed and mobile applications. Moreover, an embedding exhibits several functional advantages, such as notion of direction, or the possibility to span volumes. A more detailed discussion of these advantages is provided in Section 5.

Most state-of-the-art algorithms for graph embedding are not well suited for large graphs. Already a complexity of $O(n^2)$ exceeds memory or computation time limits. However, there exist methods that overcome these problems. Examples are the MIS-filtration algorithm of Gajer et al. [7], the high-dimensional embedding approach [9], or the landmark MDS algorithm (LMDS) [5].

We have decided to use LMDS, as it is not only fast[4] but also exhibits other appealing properties. First, the result closely resembles the widely used MDS embedding. Second, adjusting the number of landmarks follows a time-quality trade-off that allows to well adapt the resulting embedding to the application's needs. Third, LMDS behaves well in dynamic settings. New nodes can be added to the embedding by placing them according to their distances to the landmark nodes, without changing the existing coordinates.[5] We improve on the basic LMDS algorithm by introducing the idea of *iterative embedding*, which will be discussed next.

### 4.1. Iterative Embedding

*Iterative embedding* is a major ingredient to the process of mapping our music graph into a Euclidean space. The basic idea is to successively improve the embedding using a feedback loop that estimates the correctness of links using the coordinates calculated in the previous round. The basic technique applies to any sort of embedding algorithm. However, it assumes that the underlying data exhibits some randomness in its link structure, i.e., that some links erroneously shortcut certain paths. This property is generally attributed to small-world networks. Both the model of Watts and Strogatz [26] and the model of Kleinberg [10] for such graphs base on this kind of edges. We expect the music graph to exhibit such characteristics, much like other naturally grown graphs, such as social networks, the WWW, or the graph of Wikipedia articles.

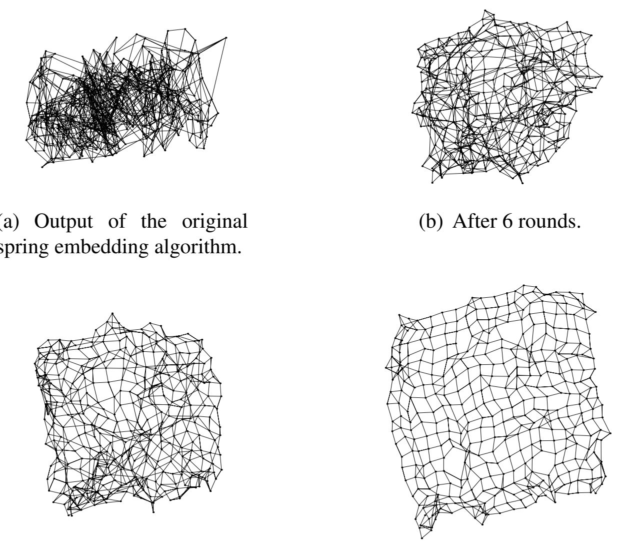
First the embedding algorithm is applied to the graph, resulting in a set of coordinates. Based on these coordinates, the fraction $f$ of edges with maximum stress is removed. For our experiments we assumed that random edges get particularly long, meaning the stress increases as the ratio $d_E/d_G$ increases (other stress functions might be defined for other settings). Identifying (and removing) this fraction $f$ of edges can be done in $\mathrm{O}(m \log(m))$ time. Next, a new set of coordinates is calculated by embedding the graph after edge removal. This process is repeated $k$ times, where $k$ should be chosen such, that $f \cdot k$ approximately matches the expected number of random edges. Moreover, $f$ should not be chosen too large, as this might result in wrong edges being removed. The effect of iterative embedding ($f = 0.3$) on a $20 \times 20$ Kleinberg graph (grid augmented with random edges) is illustrated in Figure 1.

For the Kleinberg graph we used a spring embedding method that is supposed to be well suited for small-world networks (based on ideas from [25]). However, the iterative approach is generic and works in conjunction with any embedding method. The effect of iterative embedding on the music graph is illustrated in the following section.

[3]Observe that songs are typically not direct neighbors, as the graph needs to be sparse. Non-sparse graphs, with, say $\Theta(n^2)$ edges are too big to be stored.

[4]The time complexity of LMDS is $O(nld+l^3)$, where $n$ is the number of vertices, $l$ the number of landmarks, and $d$ the number of dimensions.

[5]This only works as long as the new nodes do not significantly affect existing graph distances.



(a) Output of the original spring embedding algorithm.

(b) After 6 rounds.

(c) After 12 rounds.

(d) After 30 rounds.

**Figure 1.** *Iterative Embedding* **on the Kleinberg Graph: The original algorithm (a) cannot reconstruct the underlying grid. After 30 iterations, however, the grid structure can clearly be seen (f).**

## 4.2. Evaluation of the Resulting Map

To evaluate the quality of our embedding we used the genre information available at the *allmusic* website[6]. *Allmusic* is one of the largest music databases available on the Web. It provides a 3-level hierarchy of more than 700 music genres, as well as lists of "representative songs" assigned to each genre, as illustrated in Figure 2. This genre information is manually edited by community experts. We were able to match approximately 7000 songs between the *allmusic* and the *last.fm* databases. We used this subset of songs with known genre information in our experiments.

We define the distance $d_S$ of two genres in this hierarchy as the level of their least common ancestor (LCA). Based on the hierarchy, and this distance definition we can define two quality measures:

(1) *Distance comparison* $Q_L$: The more distant two songs are in the genre hierarchy, the larger their distance should also be in the Euclidean space. $Q_L$ thus summarizes the average similarity increase as a function of genre distance:

$$Q_L = \frac{1}{H_S} \cdot \sum_{h \in \{0 \ldots H_S - 1\}} \left( \frac{\bar{d}_{h+1} - \bar{d}_h}{\bar{d}_h} \right), \qquad (1)$$

[6]`http://www.allmusic.com`

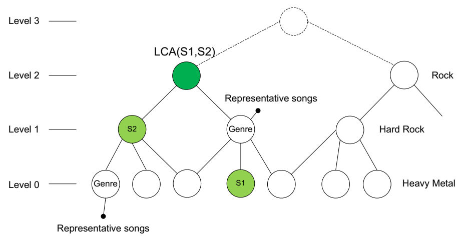


**Figure 2. Music Taxonomy: Example of a genre hierarchy. The distance between two genres is the level of their LCA. E.g. two songs $i$ and $j$ belonging to genres $s1$ and $s2$ have genre distance $d_S(i, j) = 2$.**

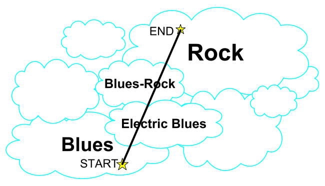


**Figure 3. Embedding smoothness $Q_R$**

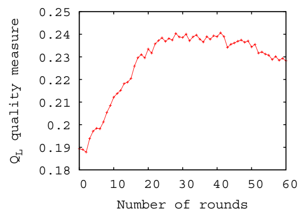
where $\bar{d}_h$ is the average similarity of pairs of items $(i, j)$ that have genre distance $d_S(i, j) = h$. We thereby assume that the similarity of unrelated songs is (typically) overestimated, such that higher values of $Q_L$ indicate better quality.

(2) *Embedding smoothness* $Q_R$: A good embedding should cluster songs of similar genres together. In particular, for each genre there should exist only one cluster, and we expect these clusters to have convex shape. Therefore, a straight line between two random points (or songs) $i$ and $j$ in the embedding should reflect a gradual and systematic genre transition from $i$ to $j$, as illustrated in Figure 3. An ideal gradual transition means that, once the line has crossed a cluster of a genre, it does not intersect the same or another cluster of this genre. The measure counts the number of violations of this rule, i.e. it counts how often an already visited genre reoccurs on a random straight line:

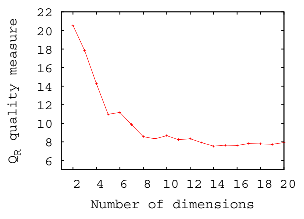
$$Q_R = \text{avg}(\textit{\#re-occurrences on a random line}). \qquad (2)$$

To implement $Q_R$, the line between two random songs is sampled at 50 uniformly distributed points. To each of these points, the genre of the closest song is assigned.

We applied iterative embedding to the music graph in conjunction with LMDS and estimated the improvement over pure LMDS using the distance-comparison quality measure $Q_L$, defined in (1). The result for an 10-dimensional embedding on 430K nodes, with parameter $f = 0.5\%$, is illustrated in Figure 4(a). The figure shows a continuous quality improvement up to approximately iter-



(a) Quality improvement using iterative embedding.



(b) Improvement of the embedding smoothness $Q_R$ with increasing number of dimensions.

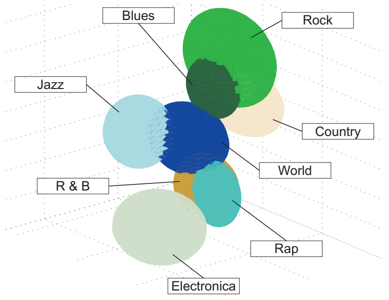
**Figure 4. Map creation: Quality analysis.**

ation 30. We expect that at this point most of the random edges have been removed. Further iterations thus result in the removal of relevant edges and hence in a reduced embedding quality.

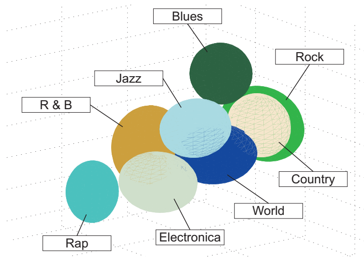
Figure 5 visualizes the main genre clusters of the embedding before and after 30 iterations. To be able to visualize the clusters, the embedding was projected into a 3-dimensional space (by taking the first three dimensions of each coordinate). The center of each cluster (ellipse) was placed at the center of mass of songs belonging to the corresponding genre. The ellipse's axes ("diameters") were set to half the value of standard deviation in each of the three dimensions. It can be seen that after the iterative embedding is applied, different clusters are better separated (even in three dimensions). In 5(a) one can see that "Blues" and "Rock", as well as "Jazz" and "World" clusters (and "R&B" and "Rap") are merged, whereas in 5(b) these pairs of clusters are clearly separated. Although this visualization is limited due to dimensionality reduction, it serves as an additional indicator of the improvement achieved with iterative embedding. Moreover, it illustrates the good overall quality of the embedding.

Note that the dimensionality of the target space significantly affects the quality of an embedding and follows a trade-off: The more dimensions the lower the distortion of the embedding becomes. However, a larger number of dimensions implies higher memory and computing time requirements for an application that operates on the coordinates. In an attempt to find the optimal number of dimensions for our purposes, we have used the embedding smoothness quality measure, defined in (2), as it best matches the objectives of our application. Figure 4(b) reveals that increasing the dimensionality significantly improves the quality up to approximately 10D. This is in contrast to previous work that focuses on embedding in at most three dimensions and aims exclusively at visualization.

Note that the axes of the resulting Euclidean space are not assigned any semantical meaning, in contrast to conventional notion of an axis being associated to some prop-



(a) Output of the original LMDS algorithm.



(b) After 30 rounds of iterative embedding.

**Figure 5. Non-iterative LMDS (a) is unable to separate the genre clusters, whereas after 30 rounds of iterative embedding, the clusters become clearly disjoint.**

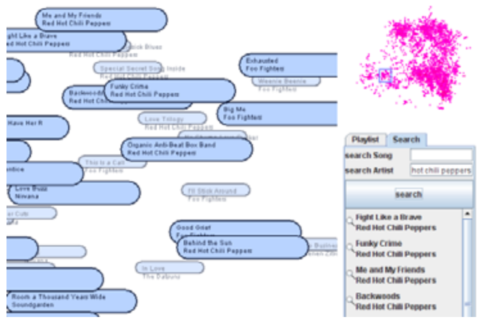
erty, such as force or time. Table 1 illustrates the 10-neighborhood of two songs. The fact that the closest neighbors of each song belong to the same artist or to very similar artists shows that (1) the co-occurrence measure is in fact able to find similar items, and (2) that the step from web to map was successful, i.e. that the Euclidean map groups similar items together.

## 5. Applications

Working on an embedding rather than on a graph exhibits several performance advantages, as has been discussed in Section 4. Applications can also profit from new functionalities, enabled by elements defined in a Euclidean space, such as *trajectories*, *volumes* and the *notion of direction*. These elements constitute the *building blocks* for an application. A good embedding places songs in the Euclidean space such that similar songs are grouped together. Therefore, regions in space can be typically associated with certain music properties, such as genre, age group, or rhythm. When a user's personal collection of songs is mapped to this "space of music", the region(s) it occupies can be compactly represented as a volume (or a union of several volumes). Similarly, a volume can be used to define the *region of interest* of a user. Trajectories, on the other hand, allow

| **Pink Floyd (Time)** | **Miles Davis (So What)** |
|---|---|
| Pink Floyd (On the Run) | Horace Silver (Song For My...) |
| Pink Floyd (Any Colour...) | Bill Evans (All of You) |
| Pink Floyd (The Great G...) | Miles Davis (Freddie Fre...) |
| Pink Floyd (Eclipse) | Nat King Cole (The More I...) |
| Pink Floyd (Us and Them) | Miles Davis (So Near) |
| Pink Floyd (Brain Damage) | Miles Davis (Flamenco Sk...) |
| Pink Floyd (Speak to Me) | Charles Mingus (Eat That Ch...) |
| Pink Floyd (Money) | Jimmy Smith (On the Sunny...) |
| Pink Floyd (Breathe) | Julie London (Daddy) |
| Pink Floyd (One of These...) | Bill Evans (My Man's Gone...) |

**Table 1. Closest neighbors of *Time* (Pink Floyd) and *So What* (Miles Davis)**



search Song
search Artist: hot chili peppers

Fight Like a Brave
Red Hot Chili Peppers
Funky Crime
Red Hot Chili Peppers
Me and My Friends
Red Hot Chili Peppers
Backwoods
Red Hot Chili Peppers

**Figure 7. *Music Explorer*: visualization of a user's collection in 2D.**

to smoothly *interpolate* between songs or regions. Finally, using sense of direction allows to *extrapolate* such trajectories. Given a sequence of songs, we can define how this list could be extended.

As a concrete example we next present *Music Explorer*—a web tool, available at *www.musicexplorer.org* that serves as a proof of concept of the presented ideas. Afterwards, we discuss several application scenarios that further motivate the ideas presented in this work.

## 5.1. Music Explorer

*Music Explorer* is an ongoing project that aims to assist users in exploring the world of music in general, and their own music in particular (see Figure 6). At the time of writing, two main features have been implemented: *playlist generation* and *visualization*. It is equally possible to operate on the entire database (i.e. the roughly 430K songs contained in the embedding), as well as on the personal collection of songs, uploaded from the user's machine.

To allow for an intuitive browsing of a collection, *Music Explorer* provides a visualization in 2D space. This space clearly exhibits a loss in accuracy as compared to the original 10D embedding (recall Figure4(b)). In particular, points that were originally far away can become close (and might even overlap) when projected into a 2D plane. However, the result is an intuitively understandable representation, which we believe justifies this loss in accuracy when it comes to merely browsing a collection.

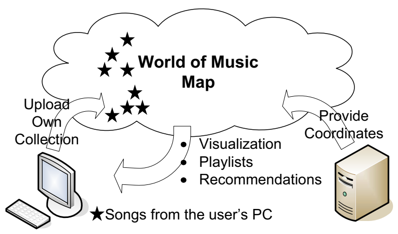


**Figure 6.** *Music Explorer*: *www.musicexplorer.org*

As a result of the applied LMDS embedding algorithm and the underlying principal component analysis, the first two dimensions of the embedding contain most information. The reduction from 10 to 2 dimensions thus becomes trivial—we take the first two coordinates of the 10D space.

To account for the restricted monitor size, we allow to zoom in and out of the map. Thereby, from top to bottom level of zooming, tracks are displayed in order of decreasing popularity. To ease navigation even at high zoom-levels, we further provide a satellite view that shows the current position with respect to the entire map. Figure 7 shows a screenshot of the visualization in *Music Explorer*.

Playlists can be created by selecting a start and an end song on the map (note that although the visualization is done in 2D, playlists are created in 10D). Playlist generation is a direct application of *trajectories*: by following a straight line between start and end songs and by considering additional constraints (e.g. distinct artists only), appropriate songs are selected, resulting in a playlist of desired size or duration. Two example playlists are shown in Table 2. It can be seen that the playlists presents a gradual transition in genre between the start points (*Milestones* by Miles Davis, and *Lose Yourself* by Eminem) and the end points (*You're Crazy* by Guns N' Roses, and *Toxic* by Britney Spears).

## 5.2. Towards a New World of Music?

*Music Explorer* is a proof-of-concept application that does not exploit the full potential of such a map. We believe that an accurate Euclidean representation of perceived music similarity can have a big impact on how people deal with music in the future. We outline some possible directions to conclude this paper.

Due to the low memory footprint, mobile environments

| | |
|---|---|
| 1. M. Davis (Milestones) | 1. Eminem (Lose Yourself) |
| 2. M. Davis (Someday...) | 2. Eminem (Stan) |
| 3. J. Coltrane (Wise One) | 3. Eminem (Mockingbird) |
| 4. R. Johnson (Kind H...) | 4. D. Bedingfield (Gotta Get...) |
| 5. J. Hendrix (Are You...) | 5. Nelly (Dilemma ft Kelly...) |
| 6. Queen (Good Old...) | 6. Pink (Most Girls) |
| 7. J. Hendrix (Gypsy...) | 7. G. Stefani (What You...) |
| 8. AC/DC (Let Me Put...) | 8. C. Aguilera (Genie in a...) |
| 9. AC/DC (Givin The...) | 9. B. Spears (Breathe on Me) |
| 10. GnR (You're Crazy) | 10. B. Spears (Toxic) |

**Table 2. Example playlists.**

are a typical example that profit from a map. Besides porting the described playlist generation technique to mobile mp3 players, our map could, for example, also facilitate an autonomous mobile file sharing application: Whenever two mobile devices come into connection range (by, e.g., Bluetooth), they exchange their users' *regions of interest* (defined as *volumes*) and exchange 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.

In an analogous manner, also peer-to-peer file sharing systems could profit from a map. Moreover, by monitoring the transmitted coordinate information, it might be possible to improve the overlay-network structure in peer-to-peer systems. An increased routing efficiency is likely to be observed if peers of similar interest (i.e. users that listen to similar music) get placed close in the overlay network.

A map might also give way for innovations at home. Instead of selecting artists, albums, or songs, users might steer their music experience with simple instructions, such as "play anything hip-hip", "not this, and not closely related songs", or "go towards Detroit house, be there in one hour".

Finally, the map of music might act as a catalyst for novel 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 receive people's favorite music by means of interest regions provided by their mobile devices. Sophisticated devices could provide even more feedback. Using motion sensors, for example, they 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.

## References

[1] J. Aucouturier and F. Pachet. Music Similarity Measures: What's the Use? In *ISMIR*, 2002.

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

[3] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. P. W. Whitman. A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures. *Comput. Music J.*, 28(2), 2004.

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

[5] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, 2002.

[6] D. P. W. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *ISMIR*, 2002.

[7] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. *Comput. Geom.*, 29(1), 2004.

[8] O. Goussevskaia, M. Kuhn, and R. Wattenhofer. Exploring Music Collections on Mobile Devices. In *MobileHCI*, 2008.

[9] D. Harel and Y. Koren. Graph Drawing by High-Dimensional Embedding. *Graph Drawing*.

[10] J. M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, 2000.

[11] 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*, 2006.

[12] M. Kuhn and R. Wattenhofer. The Theoretic Center of Computer Science. *SIGACT News*, 38(4), 2007.

[13] M. Kuhn and R. Wattenhofer. The Layered World of Scientific Conferences. In *APWeb*, 2008.

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

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

[16] B. Logan and A. Salomon. A music similarity function based on signal analysis? In *ICME*, 2001.

[17] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: an advanced social network extraction system from the web. In *WWW*, 2006.

[18] R. Neumayer, M. Dittenbach, and A. Rauber. PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *ISMIR*, 2005.

[19] E. Pampalk, S. Dixon, and G. Widmer. On the evaluation of perceptual similarity measures for music. In *DAFx*, 2003.

[20] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, 2005.

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

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

[23] J. Platt, C. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a Gaussian Process Prior for Automatically Generating Music Playlists. *NIPS*, 14.

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

[25] F. van Ham, J. van Wijk, and T. Eindhoven. Interactive Visualization of Small World Graphs. *InfoVis*.

[26] D. J. Watts and S. H. Strogatz. *Nature*.