Music Icons: Procedural Glyphs for Audio Files

Philipp Kolhoff Jacqueline Preuß Jörn Loviscach
Hochschule Bremen, University of Applied Sciences
Fachbereich Elektrotechnik und Informatik
28199 Bremen, Germany
{pkolhoff, jacqui}@fbe.hs-bremen.de jlovisca@informatik.hs-bremen.de

Abstract

Nowadays, a personal music collection may comprise thousands of MP3 files. Visualization can help the user to gain an overview and to find similar songs inside so large a set. We describe a method to create icons from audio files in such a way that songs which the user considers similar receive similar icons. This allows visual data mining in standard directory listings of window-based operating systems. The icons consist of bloom-like shapes, whose form and color depend on eight parameters. These parameters are controlled through a neural net, the input of which are audio features that are extracted algorithmically from the MP3 files. To adapt the system to the user's perception and interests, the neural net is initially trained with a small set of songs and icons. User studies done on the system demonstrate a strong perceptual relation between music and icons.

1 Introduction

Most current music content visualization methods aim at producing two-dimensional charts where each actual song occupies a certain xy position. Similar songs (where the meaning of "similarity" remains to be specified) are automatically placed close to another. Such an approach allows to visually classify music for instance according to genre. However, it breaks with the traditional display of files in standard operating systems and does not offer corresponding functions such as sorting, copying, deleting, and renaming. Furthermore, the user must rely on the automatic placement in the two-dimensional chart: The display does not offer further visual clues. However, the xy position of some songs may be grossly wrong since the reliability of automatically extracted features is limited [2].

We introduce a different approach (see Figure 1), which includes the following contributions:

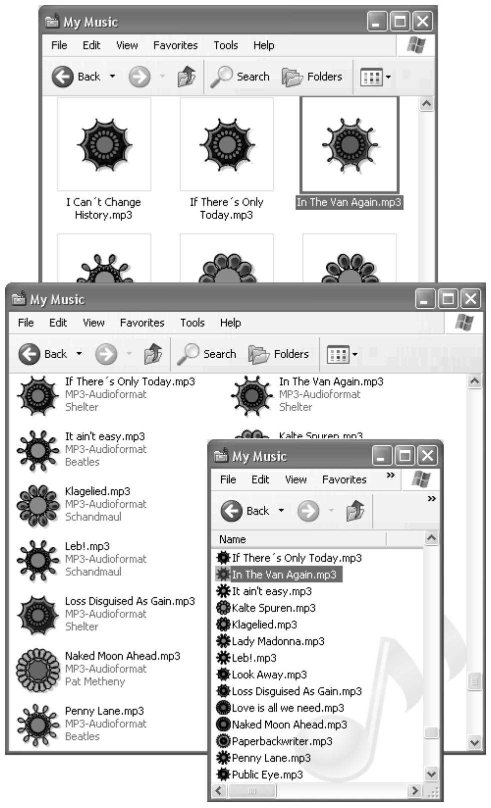
My Music
File Edit View Favorites Tools Help
Back Search Folders
I Can´t Change History.mp3
If There´s Only Today.mp3
In The Van Again.mp3
If There´s Only Today.mp3 MP3-Audioformat Shelter
In The Van Again.mp3 MP3-Audioformat Shelter
It ain't easy.mp3 MP3-Audioformat Beatles
Kalte Spuren.mp3
Klagelied.mp3 MP3-Audioformat Schandmaul
Leb!.mp3 MP3-Audioformat Schandmaul
Loss Disguised As Gain.mp3 MP3-Audioformat Shelter
Naked Moon Ahead.mp3 MP3-Audioformat Pat Metheny
Penny Lane.mp3 MP3-Audioformat Beatles
Name
If There´s Only Today.mp3
In The Van Again.mp3
It ain't easy.mp3
Kalte Spuren.mp3
Klagelied.mp3
Lady Madonna.mp3
Leb!.mp3
Look Away.mp3
Loss Disguised As Gain.mp3
Love is all we need.mp3
Naked Moon Ahead.mp3
Paperbackwriter.mp3
Penny Lane.mp3
Public Eye.mp3

Figure 1. Music Icons turn a standard file listing into a user-adaptive visualization of audio content.

- Every audio file is represented by an icon that is created automatically from the audio content of the file. This method is fully integrated with the file display in Microsoft® Windows® XP.
- A procedural, bloom-shaped pattern is employed to create the icons. This pattern may be used to visualize other multi-dimensional data, too.

- The user can specify guidelines on how the icons are to be generated. To this end, the system is trained with a handful of music files. Thus, the user's *visual* impression of the similarity of icons will match his or her *aural* impression of music similarity. For instance, one may create distinct icons for a some examples of highly specialized subgenres of electronic music, but only one icon for Country & Western.

- Since the icons display more than just two dimensions of data, the user has a better chance to deal with situations in which some features behave abnormally.

- The system is not based on an explicit knowledge of the musical or graphical meaning of the audio features and the icon parameters, respectively. Instead, the conceptual connection between music and icon exists only in the user's mind. Thus, we can create a user interface that hides all technical details concerning audio features and icon parameters.

The system can be used to find similar songs (known or novel)—or to search for music that sounds fresh and different. The user can quickly survey the content of a drive, in particular a removable disk or an MP3 player mounted via USB. There is no need to laboriously enter metadata for every song; all the user has to do is to choose icons for a handful of prototypical music files.

The method works as follows, see Figure 2: The music files are analyzed to compute features that give a concise description; a neural net maps these features to parameters used to procedurally create icons, which are handed over to Microsoft® Windows® Explorer. The prototype has been developed in C# for the .NET 2.0 platform using a number of off-the-shelf helper libraries.

This paper is structured as follows: Section 2 gives an overview on related work in icon generation and music information retrieval. Section 3 describes the procedural method employed to create icons. The audio features on which the icons are based are discussed in Section 4; the mapping from them to the icons' parameters is covered in Section 5. Section 6 gives details on the integration with the operating system. Results are presented in Section 7. Section 8 concludes, pointing out future work.

2 Related work

Lewis et al. [13] have introduced VisualIDs, the first major approach to automatic file icon generation. The names of the files are clustered; one icon is created per cluster and mutated to equip every file of the cluster with a slightly different icon. The graphical creation of the icons is based on shape grammars. In the first place, VisualIDs are intended

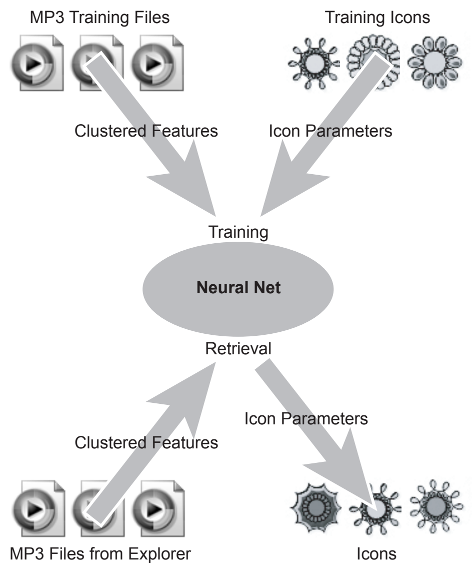

Figure 2. Music Icons rely on an association of audio features with icon parameters. This is accomplished through a neural net.

to be memorable and distinctive; reflecting the content is only of secondary concern.

The Semanticons method by Setlur et al. [22] takes more data into account, for instance abbreviations such as "fwd" used in the filename and frequent noun phrases inside text files. Based on these data, a stylized image is retrieved from a prepared collection and laid over a standard icon such as a sheet of paper with a dog's ear. In contrast to VisualIDs, Semanticons are intended to convey content. In particular, tests on the mental association between filenames and Semanticons have been performed successfully.

A standard technique in Visual Data Mining [11] is to display multidimensional data through glyphs such as Chernoff faces [6], or trees and castles [12]. More recent uses of glyphs include complex shapes to support software visualization [7] and implicit 3D surfaces [9] to be used for instance in text retrieval. Ribarsky et al. [21] present Glyphmaker, which allows the user to construct 3D shapes and create data mappings.

Music information retrieval (MIR) systems aid the user in finding music, most often music that is similar to a given song. These systems may be based on subjective data collected from users or experts, or they may rely on the wave data inside the audio files. The latter may be given in symbolic form such as MIDI files. In the general case, however, all what an content-based MIR system sees is a collection of MP3 files: sampled audio streams with no further data such as (reliable) metadata.

Typke et al. [23] give an overview of currently existing content-based MIR systems. The ISMIR 2004 Audio Description Contest [5] revealed that up-to-date algorithms yield success rates of 64 to 84 percent on the task of classifying an unknown title of music into a set of six genres from “Classical” to “World.”

The standard intermediate step to classify audio files according to their content is to drastically reduce the amount of data: Instead of the millions of original samples one uses tens of features. Typically, these are related to the spectral distribution (pitch, timbre) or to the rhythm. A selection of such features is for instance part of the MPEG-7 description [14] of an audio stream.

The standard choice for spectral features is a set of Mel-Frequency Cepstral Coefficients (MFCCs): The audio signal is sliced into “frames” of for instance 46 ms duration; these frames are subjected to a spectral analysis, which results in a logarithmic level per frequency band. The spectrum (that is, the mapping from frequencies to levels) can be smoothed to find a “spectral envelope.” MFCCs describe this spectral envelope. They result from subjecting the spectrum to an inverse Fourier transform or Discrete Cosine Transform in frequency space. Before doing so, the frequency scale is warped according to the physiologically-based Mel frequency scale.

MFCCs are computed per audio frame. A music title can be characterized by the distribution of MFCCs over all frames. Typically, this distribution is described through clusters found by k-means clustering or through a Gaussian mixture model (GMM). In such a basic approach, rhythmical properties are not considered.

Pampalk et al. [20] evaluate how well different approaches can reproduce subjective classifications of tone and genre collected from an online database. They find that MFCCs may be outperformed by a novel set of features called Spectrum Histogram. McKinney and Breebaart [16] examine another set of features: low-level properties such as pitch strength and zero-crossing rate; MFCCs; novel psychoacoustic features such as “roughness;” and temporal envelope fluctuations. All of these methods are augmented by temporal features such as the modulation intensity of MFCCs. The reliability with which a musical genre is determined remains, however, at 74 percent. Aucouturier and Pachet [2] report that current MFCC-based methods hit a “glass ceiling” at about 65 percent precision.

Currently, visualizations of music collections rely on a landscape metaphor: The songs are represented as points in 2D, where the distance between these points corresponds to the distance of the corresponding features. This requires automatic layout methods such a Self-Organizing Map or physical simulations of damped springs. For an up-to-date survey see [3]. Similar approaches have also been followed for other types of media, for instance to visually cluster thumbnails of images according to their metadata [8].

Tzanetakis and Cook [24] have introduced a number of different visualizations: TimbreGrams display the development of a song as a horizontal bar of vertical stripes, each stripe’s color encoding the features of a frame. TimbreSpace uses Principal Component Analysis to map the audio features of one frame to a point in 3D space, TimbreBall animates a 3D point according to the temporal progression of a song’s features. GenreGrams determine the confidence with which a song can be classified into twelve categories such as “Male” and “Hip-Hop.” The twelve resulting confidence values are mapped to the height of twelve 3D objects.

3 Icon generator

The simplicity of the icons is vital for searching through large numbers of files [4]. Thus, we sought an easily comprehensible and aesthetically pleasing glyph that still conveys a relatively large number of data dimensions. The mapping from parameters to appearance has to be continuous; otherwise small differences in parameters may result in overly prominent visual differences. Hence, many standard generative techniques are ruled out, such as shape grammars as applied for VisualIDs [13] or the retrieval of images from a database such as used for the Semanticons [22]. We did not aim to create abstract shapes such as VisualIDs or concrete icons such as Semanticons, but rather tried to find an familiarly-looking but neutral glyph that is visually robust enough to be used on small displays.

Our solution to this problem is a bloom-like shape. One ring of petals turned out to look overly simple; thus, we elected to create an additional inner ring of petals, whose shape parameters are coupled in reverse to those of the outer ring, see Figure 3. The inner and outer petals are formed by hypotrochoid curves. The outer curve is filled with a radial blend between two colors; the inner curve is filled with the color used for the inner part of the outer one. We employ the GDI+ application programming interface to render the icons as filled polylines with antialiasing, highlights, and a drop shadow.

The two RGB colors contribute six parameters. Four further parameters control the number and the shape of the petals. This leaves a total of eight independent parameters (six for color, one for number, one for shape) controlling the glyph.

4 Audio features

For every music file we extract the MFCC data of either the complete length or—to save time—the data of the 50th to the 59th second. To create representative audio excerpts

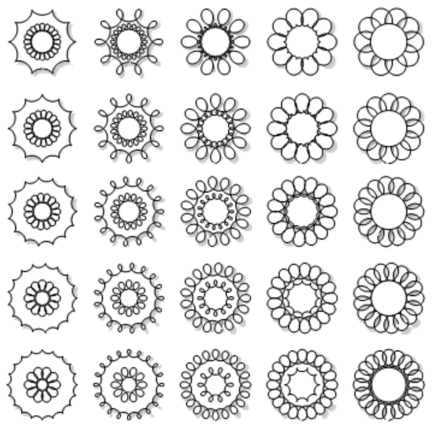

Figure 3. The geometry of the inner and outer petals is coupled to be controlled through two parameters used as x (petal shape) and y (number of petals) in this diagram.

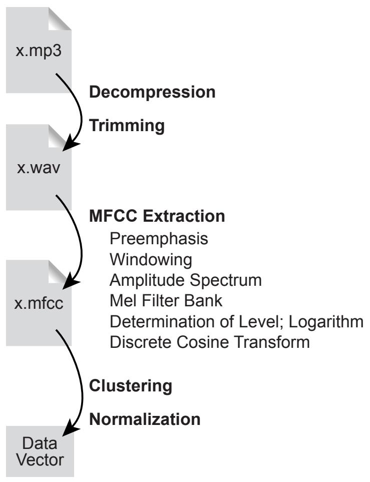

Figure 4. The feature extraction employs a pipeline partially created from off-the-shelf libraries.

a.k.a. audio thumbnails, one typically uses the time span 0:50 to 1:20 of a song. We found that even one third of this length works well for Music Icons, cf. Section 7. The parameters of the feature extraction are typical ones: frequency range 40 to 8000 Hz, eight coefficients, frame size of 1024 samples (i. e., 23 ms at a sampling rate of 44.1 kHz) with Hann windowing and 50 percent frame overlap.

To condense the sequence of MFCCs (one set of MFCCs per frame) into a concise statistical description, we form eight clusters using k-means clustering, see Figure 4. This algorithm is executed for 1000 iterations or until the clusters become stable. Generally, 30 to 100 iterations suffice, depending on whether only ten seconds or the complete music file is analyzed. The centroids of the eight clusters form the 64 final features (eight centroids times eight MFCCs) that represent every song. The eight clusters are ordered according to the number of frames they comprise, so that also the relative importance of the clusters is evident from the final collection of features. Every one of the 64 feature values is scaled and shifted so that the data from the training set cover the range from 0.0 to 1.0. This normalization simplifies further processing steps.

To read uncompressed audio data from MP3 files, we employ the madlldlib [1] wrapper of the libmad library [26]. For the extraction of MFCCs we rely on a C library contributed by Sven Fischer from Fachhochschule Oldenburg. Other off-the-shelf options for feature extraction such as Marsyas [25], the MPEG-7 reference implementation [18] for Matlab, or JAudio [15] turned out to be too heavyweight or to be too difficult to integrate. Furthermore, we wanted to first concentrate on a homogeneous set of features such as MFCCs.

5 Mapping features to shapes

At the heart of our method lies a neural network that maps the 64 audio features of a song to the eight parameters of its icon. Initially, the network is trained to suit the user's preferences: The user chooses a representative selection of music and defines an icon for every single of these songs. Later, the network is applied to the audio features of a novel song and produces the parameters of a corresponding icon.

To set the parameters of such an icon, the user is not confronted with the internal parameters. Instead, the system displays 15 preset icons created with extreme parameter settings and 15 random variations of the currently selected icon (preset or already mutated), see Figure 5. "Genetic" variations are a standard parameterless user interface for generative art, see for instance [17]. The preset icons as well as the mutation operation are biased toward extreme values of the parameters, so that the apparent differences of the icons are maximized.

The neural net is realized using an off-the-shelf library [10]. It consists of an input layer of 64 neurons and an output layer of 8 neurons. The activation function has a sigmoid shape. The training is accomplished by backpropagation; 50 iterations (which can be computed in a fraction of a second) suffice to achieve a stable output.

6 Integration with the operating system

We have developed a stand-alone software and an icon handler shell plug-in for Microsoft® Windows® Explorer, which operate together. The standalone software is used to

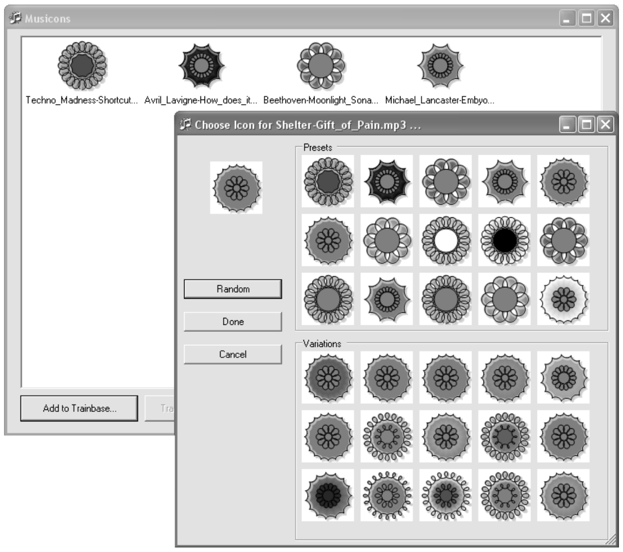
Musicons
Techno_Madness-Shortcut...
Avril_Lavigne-How_does_it...
Beethoven-Moonlight_Sona...
Michael_Lancaster-Embyo...
Choose Icon for Shelter-Gift_of_Pain.mp3 ...
Presets
Random
Done
Cancel
Variations
Add to Trainbase...

Figure 5. During the training, the user has to specify basic icons. To this end, our software offers a set of presets and a set of genetic variations.

select the training music as well as the corresponding icons and to train the neural net. On demand, the software creates icons for a list of MP3 files: It extracts the features, uses the previously trained neural net to generate the icons' parameters, and draws the icons into bitmaps of the four standard sizes 16×16, 32×32, 48×48, and 64×64 pixels. The software attaches a proprietary ID3 tag [19] to store such a suite of four icons inside every processed MP3 file. This allows a simple management of the icon data as opposed to a special file cache. The embedded icons lead to only a negligible increase in the files' size: A typical file will be several megabytes long, with the embedded icon contributing 30 kilobytes.

If the user opens a directory in Explorer that contains MP3 files, the shell plug-in looks for embedded icons inside every contained file. If no icon is found, a placeholder is displayed; in this case, the file is added to the list of MP3 files to be subjected to icon generation by the standalone software. Our proprietary ID3 tag contains an identifier that allows to treat the icon as missing if the current neural net has been trained differently.

7 Results

On an AMD Athlon™ 64 3000+ running at 1.81 GHz, to generate icons for a song of four minutes' length takes 3.7 s for the decompression of the MP3 file into a WAV file, 6.4 s to extract the MFCCs, 11.0 s to cluster them, and 0.17 s to generate and embed the icons. If only ten seconds of the music file are used, the times to extract and to cluster the features reduce to 0.27 and 0.04 s, respectively. To trim the WAV file to ten seconds incurs an overhead of 0.16 s, so that one MP3 file can be processed completely in less than four seconds. This speed gain does not lead to vital changes in the icons' appearance, see Figure 6.

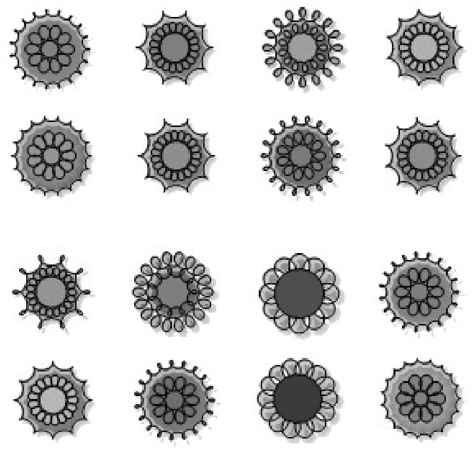

Figure 6. Icons generated from complete MP3 music files (lines 1 and 3) do not differ drastically from icons generated from ten-second excerpts (lines 2 and 4).

To optimize the extracted features, we varied the settings systematically: 5, 8, 13, and 19 MFCCs; 2, 4, 8, and 16 clusters being formed; the most prominent 2, 4, 8, and 16 of these clusters being fed into the neural net; 3 to 18 parameters used to generate an icon. The final settings of 8 MFCCs, 8 clusters being both formed and used, and 8 parameters for the icons resulted from these experiments.

If one looks at the relation between artists and icons, the results (see Figure 1) are immediately compelling. We wanted to gain more insight on this relation and thus conducted two tests with prospective users of the system. The first test was on quantitative aural and visual distance, the second test concerned similarity/dissimilarity. For the tests we trained the system with five songs and icons and created Music Icons for 64 other music titles from classics, folk, techno/trance, rock, pop, and punk.

Our first hypothesis was that the user has a mental model of "distances" in a perceptual "space of icons" and a perceptual "space of songs" and that distances in these spaces are related. To test this hypothesis, we asked seven users (age about 25; four female, three male) to specify for each of the 64 songs how well they match the five songs used for training. To learn about the user's notion of what constitutes "likeness" of music, we took care to not specifically ask for a certain sort of likeness. For every single of the 64 test songs, the users had to assign a mark from the Likert scale $--$, $-$, 0, $+$, $++$ to every of the 5 training songs. The users had to fill in a similar questionnaire for the 64 generated icons with reference to the training icons. Treating the Likert scale not as ordinal data, but approximately as interval data, we mapped the Likert scale to the range of numbers from -2 to 2 and computed the Pearson correlation coefficient between the data collected for the icons and for the music. Depending on the user, this was in the range of 0.45 to 0.55. Thus, the results from the first test signify a certain effect, but not an effect as drastic as a quick glance onto the results in Explorer seems to indicate.

The effect may be limited due to several reasons: Likeness of music is a very subjective notion which refers to acoustic features as well as the epoch, the text, the performer, and the listener's biography. On top of that, the mental model of music and that of imagery may not resemble linear space. It may rather behave like a strongly curved manifold or like a network, so that it's impossible to reasonably specify if a song A is 50 percent of a song B plus 30 percent of a song C plus 20 percent of a song D.

Thus, we came up with a second, more general hypothesis: The user has an internalized notion of similarity of timbre and a internalized notion of similarity of (abstract) icons, and both notions can be mapped onto each other. To test this hypothesis, to more closely model an actual task, and to circumvent many of the issues found in the first test, we designed a different experiment: Using the same song and icon material as before, we asked test persons to visually form clusters of the 64 icons, without knowing song titles or other information. They could use any number of clusters they considered appropriate and were allowed to leave a set of spare icons that don't fit to anything else. Figure 7 indicates that the test users agreed strongly about visual likeness. The differences point toward users applying different thresholds for what constitutes "similar" icons.

As Figure 8 demonstrates, the icon clusters formed this way appear to be very reasonable if one looks at the artists' names, which were hidden from the test users. Titles of a single album were only rarely scattered over several clusters. But even in these cases the different assignments seemed to make sense because the styles of the songs were indeed drastically different. To gain quantitative evidence, we tested whether the icon clusters formed visually by a user corresponded to music clusters in this user's perception. To this end, we selected three or four songs from one cluster by random, added one song from another cluster by random and offered these songs to the user in a random sequence. The user was asked to spot the song that sounded different from the others.

If there was no relationship at all, the probability to spot the song from the other cluster would be 0.25 or 0.2, depending on whether we offered four or five songs. However, the probability measured in our experiments was consistently well above 0.5, see Table 1. This rate seems to be close to what can be hoped for, given the current recognition rates found in content-based music retrieval systems, see Section 2. The data even lead us to suspect that the recognition rate improves if *more* songs are presented. Here, the chances to pick the single mismatching song by random are even smaller; however, we suppose that the larger number of music files makes it easier for the user to understand the pattern—to learn what the clusters he or she has formed visually mean acoustically.

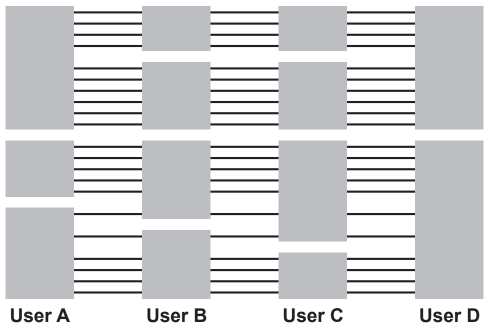

Figure 7. The users form similar clusters from the icons. In this excerpt of the data, the horizontal lines represent songs and the blocks represent clusters. The order of the songs has been adapted to reduce visual clutter.

8 Conclusion and outlook

We have presented a complete system to support searching for similar music or for different music inside standard file views with all their amenities. It is based on a synaesthetic relation between icons and timbre. The software does not form clusters of songs. Rather, it offers clues to the user, adhering to the spirit of visual data mining. The clues vary smoothly with the underlying features but behave in a qualitative manner. They do not resemble a classic measurement device with a clear scale ranging from 0 to 100. As our tests with prospective users demonstrate, the visual clues we chose carry great acoustic importance. They do not only allow to distinguish between genres, but also between subtypes of a given genre, such as classical music performed on a piano or by a full orchestra.

The system adapts to the user in that he or she can select training music and training icons. Thus, it can cater for narrow selections of music and can take the visual perception of the user into account. For instance, spiky petals may be used for harsh sounds; a color-blind user may create icons with strong contrast in brightness; a user suffering

User	Clusters Formed	Choice Presented	Recognition Rate
A (male, 30 y)	23+19+10+6+5, unassigned: 1	1 of 4	8 of 15
B (male, 27 y)	27+15+6+5+5+4+2	1 of 4	6 of 15
C (male, 37 y)	23+13+7+6+4+4+2, unassigned: 5	1 of 5	8 of 11
D (male, 55 y)	26+17+11+10	1 of 5	10 of 12
E (female, 26 y)	41+5+4+7+7	1 of 5	6 of 7
F (female, 23 y)	22+9+6+5+4+4+3+2+2+2, unassigned: 5	1 of 5	9 of 11

Table 1. The users had to visually cluster the icons (Clusters Formed), and were then presented a series of playlists in which they had to spot the song that sounds different from a set of four or five (Choice Presented). The number of successful attempts was counted (Recognition Rate).

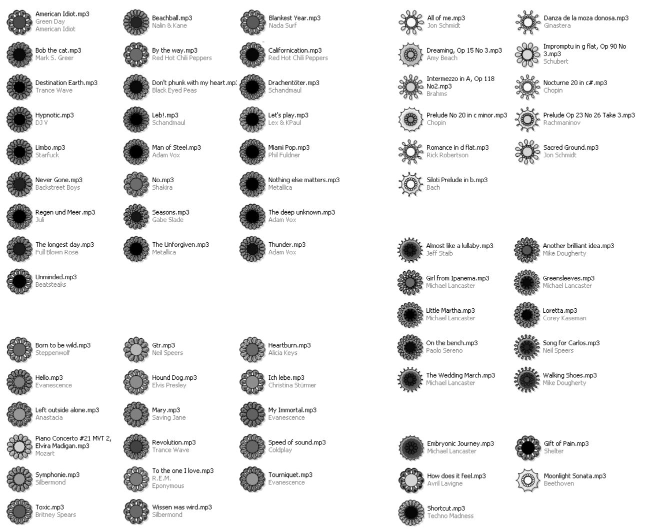

Figure 8. The test users (in this case: user D) had to cluster the icons visually. They did not see the titles of the songs nor did they see the training songs, which are given for reference in the lower right of this diagram.

from synaesthesia may associate soul ballads with the color orange. Such personalized settings should result in even higher recognition rates.

Adaptivity also works in the other direction: The user learns to "read" the icons in terms of music. In particular, our tests with smaller and larger example sets for clusters of songs indicate that users can understand the technical implementation of acoustic "similarity." This turns the vague concept of "music like ..." into an algorithmic definition. Actually, what the phrase "something sounds similar" means in terms of human perception is very hard to define. Similarity is also difficult to define for icons. This is why we elected to do user studies: We wanted to learn if our technical implementation reflects crucial aspects of human perception—which it obviously does.

On the technical side, to extract features and generate icons is ideally suited as a background task for a multi-core processor. The speed of the extraction can become an issue if one plugs in a memory device with thousands of music titles on it. In this case, one needs to spend some minutes on some other task and then may return to the music files, now with icons. Currently, we are working on a two-level approach where the icon generator first creates icons quickly based on a ten-second segment of every song (possibly operating directly on the compressed MP3 data or using an MP3 splitter tool to extract the ten-second portion); later the icon generator processes the full length of every song to replace the quick-and-dirty icon it has generated before. Furthermore, we envision a plug-in architecture for both icon generation and feature extraction, similar to the visualization plug-ins found in current music player software.

To also display Music Icons for files on read-only disks, one may create a proprietary icon cache on some system disk instead of embedding the icons to the MP3 files. In total, however, the embedded icons offer great advantages over a central icon cache. For instance, MP3 files with embedded icons can also be used on devices with low computing power such as mobile phones and PDA computers. We aim at providing corresponding extensions to the file browsing software of such devices.

Whereas Music Icons are currently based on spectral features, future work may address rhythmic features as well. Applications such as DJing may require to find songs with similar tempo or with similar rhythmic patterns.

Acknowledgments

P. Kolhoff and J. Preuß have been supported by Microsoft® Germany. J. Loviscach's work was partially funded by grant 1742A04 of the German Ministry of Education and Research (BMBF). The views and conclusions contained in this document are those of the authors.

References

[1] Arbinger Systems. DLL to decode MP3 to WAV/PCM. http://www.codeproject.com/audio/madlldlib.asp, 2004.

[2] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[3] S. Baumann. Visualization for music IR. ISMIR 2005 Tutorial, 2005.

[4] M. D. Byrne. Using icons to find documents: simplicity is critical. In *CHI '93: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 446–453, 1993.

[5] P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack. ISMIR 2004 audio description contest. Music Technology Group, Universitat Pompeu Fabra, Technical Report MTG-TR-2006-02, 2006.

[6] H. Chernoff. Using faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[7] M. C. Chuah and S. G. Eick. Glyphs for software visualization. In *WPC '97: Proceedings of the 5th International Workshop on Program Comprehension (WPC '97)*, pages 183–191, 1997.

[8] M. Dontcheva, M. Agrawala, and M. Cohen. Metadata visualization for image browsing. UIST 2005 Demonstration, 2005.

[9] D. S. Ebert, R. M. Rohrer, C. D. Shaw, P. Panda, J. M. Kukla, and D. A. Roberts. Procedural shape generation for multi-dimensional data visualization. In *Data Visualization '99*, pages 3–12, 1999.

[10] F. Fleurey. C# neural network library. http://franck.fleurey.free.fr/NeuralNetwork/index.htm, 2002.

[11] D. A. Keim, M. Sips, and M. Ankerst. Visual data-mining techniques. In C. D. Hansen and C. Johnson, editors, *Visualization Handbook*, pages 831–843. Academic Press, Burlington, MA, 2004.

[12] B. Kleiner and J. A. Hartigan. Representing points in many dimensions by trees and castles. *Journal of the American Statistical Association*, 76:260–269, 1981.

[13] J. P. Lewis, R. Rosenholtz, N. Fong, and U. Neumann. VisualIDs: automatic distinctive icons for desktop interfaces. *ACM Trans. Graph.*, 23(3):416–423, 2004.

[14] J. M. Martnez, R. Koenen, and F. Pereira. MPEG-7: the generic multimedia content description standard, part 1. *IEEE MultiMedia*, 9(2):78–87, 2002.

[15] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. JAudio: a feature extraction library. In *ISMIR 2005: Proceedings of the 6th International Conference on Music Information Retrieval*, pages 600–603, 2005.

[16] M. F. McKinney and J. Breebaart. Features for audio and music classification. In *ISMIR 2003: Proceedings of the 4th International Conference on Music Information Retrieval*, pages 151–158, 2003.

[17] J. Meyer-Spradow and J. Loviscach. Evolutionary design of BRDFs. In *Eurographics 2003 Short Paper Proceedings*, pages 301–306, 2003.

[18] MPEG. MPEG-7 eXperimentation Model. http://www.lis.ei.tum.de/research/bv/topics/mmdb/e_mpeg7.html, 2002–2005.

[19] M. Nilsson and J. Sundström. ID3v2. http://www.id3.org/, 1998–2005.

[20] E. Pampalk, S. Dixon, and G. Widmer. On the evaluation of perceptual similarity measures for music. In *DAFX-03: Proceedings of the 6th Int. Conference on Digital Audio Effects*, pages 7–12, 2003.

[21] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualizations of complex data. *Computer*, 27(7):57–64, 1994.

[22] V. Setlur, C. Albrecht-Buehler, A. A. Gooch, S. Rossoffa, and B. Gooch. Semanticons: Visual metaphors as file icons. *Computer Graphics Forum*, 24(3):647–656, 2005.

[23] R. Typke, F. Wiering, and R. C. Veltkamp. A survey of music information retrieval systems. In *DAFX-05: Proceedings of the 8th Int. Conference on Digital Audio Effects*, pages 153–160, 2005.

[24] G. Tzanetakis and P. Cook. 3D graphics tools for sound collections. In *DAFX-00: Proceedings of the COST G-6 Conference on Digital Audio Effects*, pages 115–118, 2000.

[25] G. Tzanetakis and P. Cook. Marsyas: a framework for audio analysis. *Organized Sound*, 3(4):169–175, 2000.

[26] Underbit Technologies. MAD: MPEG audio decoder. http://www.underbit.com/products/mad/, 2004.