DESIGN AND EVALUATION OF A VISUALIZATION INTERFACE FOR QUERYING LARGE UNSTRUCTURED SOUND DATABASES

Frederic Font Corbera

MASTER'S THESIS UPF / 2010

Master in Sound and Music Computing

August 10, 2010

Gerard Roma, Jordi Janer - Supervisors

Department of Information and Communication Technologies

Universitat Pompeu Fabra, Barcelona

Abstract

Search is an underestimated problem that plays a big role in any application dealing with large databases. The more extensive and heterogeneous our data is, the harder is to find exactly what we are looking for. This idea resembles the *data availability paradox* stated by Woods [45]: "more and more data is available, but our ability to interpret what is available has not increased". Then the question arises: is it really useful to collect a big dataset even if we do not have the ability to successfully navigate among it?

According to Morville and Callender [27], search is a grand challenge that can be succeeded with courage and vision. A good searching tool completely improves the exploitation we can do of our information resources. As a consequence, commonly used search methods must evolve. Search goal is more than finding, search should become a conversation process where answers change the questions.

Having stated all that, it seems clear that extensive effort should be invested on the research and design of appropriate tools for finding our needles in the haystack. However, search is a problem that does not have a general solution. It must be adapted to the context of the information we are dealing with, in the case of the presnet document, unstructured sound databases.

The aim of this thesis is the design of a visualization interface that let users graphically define queries for the *Freesound Project* database (`http://www.freesound.org`) and retrieve suitable results for a musical context. Music Information Retrieval (MIR) techniques are used to analyze all the files in the database and automatically extract audio features concerning four different aspects of sound perception: *temporal envelope*, *timbre*, *tonal information* and *pitch*. Users perform queries by graphically specifying a target for each one of these perceptual aspects, that is to say, queries are specified by defining the physical properties of the sound itself rather than indicating its *source* (as is usually done in common text-based search engines). Similarity search is performed among the whole database to find the most similar sound files, and returned results are represented as points in a two-dimensional space that users can explore.

Acknowledgments

First of all, I would like to thank my supervisors Gerard Roma and Jordi Janer for their support and advice during the development of the present master thesis and for all the time that I took from them. I would also like to thank Perfecto Herrera for helping me whenever I asked (specially during the evaluation process of the interface) and Nicolas Wack for answering all my silly questions regarding Gaia.

I would also like to thank all my classmates of the SMC master, which have made this experience even richer. Specially to Fran, Laia, Pablo and Zuri, that have been here since the first day, and have helped me whenever *existential* doubts have appeared (and let me help them as well).

Finally, I would like to thank all my family and friends, that have been always around: Lurdes, Pere, Eduard, Nicolau, Aleix, Joan(s), Antoni and Martí. And very specially to Anna, who has closely lived this process and never complained, encouraging me whenever I needed.

Frederic Font Corbera
Granollers, August 2010

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, never-ending increase of multimedia content sharing over the net has led to a point where practically *everything* is accessible at any time. With audio being one of the most popular types of online information, millions of sound recordings are stored in the net. However, which are the techniques available to navigate among this ocean of information? How can we find the recordings we have in mind? Lets focus in the case of sound (not music) seeking. Usually, we are given an empty search box that is used to input textual concepts describing a sound. Later, this information is used to query a database, looking for entries whose *tags* or *labels* resemble the description. In some cases this approach is quite efficient, but requires a properly labeled database and, what is even more important, a well defined target for the search, that is to say, that users know exactly what they are looking for and how to textually describe it. Even when a textual description can be formalized, users are inevitably constrained by the ambiguity and informality of the natural language. Moreover, the quality of the results obtained using this approach, also depends on the quality of the tags in the database. As databases are quickly increasing in size, it is harder to maintain a good and reliable manual tagging of their content, thus adding more problems to the current scenario. Considering all these points, it seems clear that we must think of techniques for aiding the browsing of large sound collections.

Music information retrieval (MIR) is a field that has been intensively researched in the last decade and that has provided new strategies for approaching this problem. MIR techniques explore content information (audio waveforms themselves) to provide automatic descriptions of the nature of sounds. These descriptions range

from *low level* features such as pitch or temporal and spectral characteristics, to *mid level* properties such as tonality, chord progression, rhythmic pattern or source identification (instrument recognition), and finally to *high level* descriptions such as evoked emotions or moods. The ability to extract these descriptors has allowed us to build feature spaces where sound files can be compared, thus providing new ways of navigating audio collections by finding *similarity* between sounds (without the need of tags) or using *query-by-example*. A direct application of this concept has been used in techniques such as *Musical Mosaicing* [49] or *Concatenative Sound Synthesis* [37], where search targets are based on extracted audio features.

Considering that feature extraction can be done automatically, some studies have suggested the use of MIR techniques to systematically label sound databases. This includes work done in [9], where non-labeled sounds are used as target for similarity search in labeled databases, and tags commonly found in the results are adopted. WordNet ontologies are used to semantically organize the categories that comprise different tags, in such a way that by having obtained the tag *car* for a sound, we automatically know that it also belongs to a higher category named *vehicle*. This allows us to minimize in some way the problem of natural language ambiguity, and provides a partial solution for the manual tagging issue. However, the problem of knowing how to textually describe the sounds we are looking for still remains.

Considering this issue, we think that a completely different approach than that of the traditional text-based search can be taken, focusing on the concept of querying databases by indicating the perceptual qualities of a desired sound rather than its source. Some interest in a similar concept has been shown in works such as [33, 3, 6]. The idea of giving importance to the perceptual qualities regardless of the knowledge we have about the *source* of sounds, is that of not describing a sound with something like *"a door closing"*, but rather use concepts such as *"loud, bright and impulsive"*. This gives us the possibility to describe both natural and *abstract* sounds that do not have identifiable sources.

1.2 Orientation

Considering that the main point of the present thesis is to take an approach in sound searching based on the perceptual qualities of sound, one of the first things we started to think was *how* would users feed this information. Using MIR techniques we can extract a lot of descriptors that resemble (at different levels) a variety of aspects of sound perception, so we could just let users introduce a query target by specifying the values of these descriptors. This seems to be clear as a possible

approach (another one would be to use tags describing sound perceptual qualities). However, considering that there are many aspects of sound perception that could be described (and we can extract lots of features), we need to specify what do we mean by "perceptual qualities of sound" in general. A first consideration is that the collection of sounds we can find in an online database can be so heterogeneous that it is quite difficult to try to define a general way to describe sound perceptual qualities that could fit all sounds. Actually, some studies have been done in the direction of defining a morphological sound description framework (briefly reviewed on section 2.1), but either they are too theoretical or too focused on specific kind of sounds. In [33], we can find a proposed computational model of Schaeffer's typo-morphology. Some low level features are extracted and mapped through a classifier (either directly or as a combination of features) to different characteristics of the typo-morphology. Nevertheless, we felt with the need of simplification and decided to orientate the thesis for a somehow specific subset of sounds.

One of the ideas shown in our first proposal was that we wanted to be able to exploit the content of huge sound databases in creative ways. Therefore, we decided to design the interface aiming at the retrieval of sounds that can fit in the context of musical compositions or performances (that is to say, to use relevant perceptual qualities in a musical context). To take this decision, we also got inspired by the *Freesound Radio* [34] project which, in our opinion, confirms the creative potential that stands behind huge online sound databases for music creation.

The last big consideration to define the orientation of the present thesis, is the use of a visual interface instead of traditional text-based (both for defining the query and representing the results). Assuming that we have a small list of relevant descriptors to retrieve sound files for a musical context, we still think that users need something different than traditional *input boxes* or *sliders* to introduce them. That is why we thought of an editable graphical representation of audio features that could serve at the same time to define the queries and to explore results without listening to them (up to some extent). Visual query definitions have been successfully applied in other contexts such as text-based boolean queries [22], but also in the context of music databases, as seen in *SonoSketch* project [3]. In the other hand, regarding the graphical representation of the results, we have seen that visual data exploration is known to be particularly useful when little is known about the data and the exploration goals are vague [23]. From that point of view, graphically displaying the results of a query has its advantages and reinforces the idea of *discovery* of the database in contraposition to only *search*. Moreover, information visualization techniques for representing sound collections have been successfully applied in a number of projects such as [6, 29, 20].

Chapter 2

Literature review

In this chapter, we can find a review of a variety of systems and technologies that are related to the approach we have chosen for our search interface. It is organized in sections according to different research fields that play an important role on our proposed system.

2.1 Perceptual sound description

Some attempts have been done with the aim to define a framework for sound description. By *sound description* we refer to the identification of the qualities of a sound that we perceive when listening to it. In general terms, we can divide these qualities in the information given by the identification of the *source* (that is to say, the object that causes a sound) and the physical properties of the sound itself. We refer to this second category as *perceptual sound description*. When describing sound perceptually we focus on characteristics such as the dynamics, the timbre or the pitch of the sound, rather than its meaning as a consequence of some event. These descriptions are specially interesting when we can not describe a sound by its source, either because we are not interested in the source or because it is abstract or poorly defined.

In this direction, Schaeffer established the previously mentioned typo-morphology for general sound description [31]. This morphology identifies the qualities of the sound that are perceptually relevant and organizes them according to different categories. The *matter criteria* represents, in general terms, the spectral distribution of the sound. *Shape criteria* is used to describe temporal characteristics such as the dynamics or amplitude modulations. Finally, *variation criteria* explains changes in both matter and shape. Table 2.1 (taken from [8]) shows the general scheme of

Schaeffer typo-morphology.

From a less theoretical and a bit more practical point of view, Grey performed similarity experiments between 16 musical instrument tones trying to devise the most important perceptual characteristics that explained their relationships [19]. As a result, he found a three dimensional timbre space where the first axis seemed to be closely related with some kind of *spectral energy distribution* and the other two connected to diverse temporal and spectral features of the tones. Although the results are not much clear, it is interesting as an attempt to identify perceptually relevant characteristics of sound using a practical approach (in contrast with Schaeffer).

It is also interesting the work done by Slawson regarding the *color* of sound [40]. He refers to *sound color* as an abstract property of an auditory sensation. Recalling the *source-filter* model for sound production (as also done in the explanation by the author), color is everything given in the filter stage, aside from the source or excitation part. With the concept defined, he started searching what he called the *dimensions* of sound color. In the specific case of vowel sounds, he found that they can be described with three dimensions: *acuteness*, *openness* and *laxness*. The first two are correlated with first and second formants center frequency of the spectrum, while the third dimension is a mixture of previous two. In that case, the results of the study are a bit more specific than the previously presented investigations, although dimensions are only calculated for vowel sounds and could hardly fit in other kind of sounds.

In [15] Gaver introduced a taxonomy to describe environmental sounds. He defines a classification of basic sound events including *vibrating objects*, *aerodynamic sounds* and *liquid sounds*, and the interactions that can cause solids (vibrating),

MATTER CRITERIA		
MASS Perception of "noiseness"	HARMONIC TIMBRE Bright/Dull	GRAIN Microstructure of the sound
SHAPE CRITERIA		
DYNAMICS Intensity evolution	ALLURE Amplitude or Frequency Modulation	
VARIATION CRITERIA		
MELODIC PROFILE: pitch variation type	MASS PROFILE Mass variation type	

Table 2.1 – Schaeffer typo-morphology.

gasses (aerodynamic) or liquids to sound. Furthermore, these interactions are related to specific effects than can have in the produced sound wave, both in temporal and frequency domain.

2.2 MIR and sound description

MPEG-7 and Cuidado [30] projects comprise a variety of commonly used sound descriptors in an effort of standardization. These descriptors mostly consist on low-level and mid-level features describing temporal shape (envelope), energy content, spectral characteristics (related to timbre), harmonic features (*i.e.* harmonic/noise ratio, harmonic deviation) and basic perceptual features (computed using a model of human auditory system).

More advanced mid-level to high-level audio descriptors have also been developed and are nowadays an important focus of the MIR research. Those include concepts that are more meaningful to human perception than low-level descriptions. Successful examples of these attempts include a variety of algorithms for, just to name a few, pitch detection (a review of them can be found on [7], with a special mention to the YIN algorithm [14]), melody extraction [32], tonal information extraction (for chord progressions or key detection [16]), and for rhythm description [18]. Even higher-level descriptions for sound and music have been developed for genre or mood classification. These approaches use trained classifiers based on low and mid-level features, trying to devise which descriptors are perceptually more relevant for humans when identifying genres or moods. Current state of the art of these techniques can be reviewed on [35] and [25] respectively. Table 2.2 (taken from [11]) shows an overview of currently researched MIR tasks that include descriptors above mentioned plus some others, giving an idea of current trends.

2.3 MIR and sound databases

Musical mosaicing (as presented in [49]) is one of these applications where MIR techniques are used to query a database. As an analogy to image mosaics (where a bigger image is composed by assembling a large number of smaller ones), the idea behind this technique is to create sequences of short audio segments. In order to do that, a target is defined by imposing constraints for each short segment of the mosaic to be built. Each constraint sets a target value for one content-based descriptor. These descriptors resemble audio characteristics like pitch, loudness or timbre information. As a consequence, each piece in the mosaic can be defined by

High-level Description	Data Source	Task Description
Timbre	Audio	Instrument Recognition Percussive, Pitched, Ensemble Recognition
Melody / Bass	Audio / Symbolic	Melody-line extraction Bass-line extraction
Rhythm	Audio	Onset detection Meter identification Meter alignment (bars) Beat (tactus) tracking Tempo tracking Average tempo
Pitch	Audio	Single fundamental freq. Multiple fundamental freq.
Harmony	Audio / Symbolic	Chord label extraction Bass-line extraction
Key	Audio / Symbolic	Modulation tracking Pitch spelling
Structure	Audio / Symbolic	Verse / chorus extraction Repeat extraction
Lyrics	Audio	Singing detection, lyrics-identification, word recognition
Non-Western music	Audio	Micro-tonal tuning systems Non-Western canon of concepts

Table 2.2 – Overview of currently researched higher-level MIR tasks.

a vector of target values for audio features, and a database of samples is queried in search for the best fitting sample for each position of the sequence. This is the *unit selection* process, and it is performed by computing similarity between the features of each sample in the dataset and the constraints of the target. Usually, the target sequence is constructed by extracting the features of an audio file which is not present in the dataset. Therefore, at the end we obtain a new audio file that has very similar features as the target but that is constructed by combining small samples from other files. In musical mosaicing, user is not directly facing the problem of querying the huge dataset, but the system is doing that work using MIR techniques.

A similar idea to the music mosaicing is the one explored in the *Caterpillar* system for *data driven-concatenative sound synthesis* [36]. In that case, the target is defined by a symbolic score, and the source database contains small segments of different sounds (traditionally, recorded instruments played with different expressive qualities). The unit selection process takes into account the similarity of each sample with the target (according to descriptors), but also a *concatenation quality* factor

Figure 2.1 – *SonoSketch* allow users to "draw" sound atoms onto a pitch/time grid.

that resembles how well two consecutive samples can fit.

Some other interesting systems using MIR techniques to extract information from databases include the *Islands of Music* [29], the *Sonic Browser* [6] or the more recent *SoundTorch* [3] and *SonoSketch* [20].

On *Islands of music*, similarity between pieces of music is estimated based on psychoacoustic models. Features analyzed compress spectrum and periodicity histograms that are later used to extract timbre and rhythm patterns. These features are used in conjunction with textual metadata such as artist or genre. Finally, pieces are organized on a 2D map so that similar ones are located close to each other, creating *islands* of music that resemble genres or styles. On the other hand, *Sonic Browser* uses different descriptors that range from basic audio features such as file size or sample rate (which do not require MIR analysis), to content-based ones like pitch, beat strength, Mel Frequency Cepstrum Coefficients (MFCC) or Linear Prediction Coefficients (LPC) to display an audio database in a variety of representations. In a similar way, *SoundTorch* uses MFCCs to organize a dataset on a 2D map according to psychoacoustic similarity.

Finally, *SonoSketch* presents a novel interface where users define queries for a sound database by *sketching* sounds. The idea is that user is provided with a variety of drawing tools that represent basic atoms of a sound and that can be placed onto a pitch/time grid. This sketch is synthesized as an audio file by adding the waveforms corresponding to all single atoms. MFCCs and chroma features are extracted from the resulting audio file and used as a target set of descriptors to search in a database. Figure 2.1 shows a screenshot of the drawing interface. This project served as an important source of inspiration for the present proposal by means of reinforcing the idea that graphical queries are an interesting idea to explore. Nevertheless, we will see later that, regarding important basic concepts, our approach strongly differs from that one taken in this project.

2.4 Information Visualization

How we see what we have found? The way in which search results are presented to the user has a big impact in determining the effectiveness of a search process. Morville and Callender state that Search Engine Results Page should be designed with the aim of provoking exploration, insight and understanding to the users [27]. They also suggest that we have finally seen some real improvements on that field by making use of information visualization techniques.

Information visualization is a research field that had grown tremendously during the 90's and up to nowadays. The aim of information visualization is to generate abstract visualizations of information that can reveal insights on the data by optimizing the use of human perceptual visual-thinking ability [12]. By transforming abstract phenomenon into visual forms and displaying its relationships (*i.e.* representing the contents of a database in a search context), users can be directly involved in an exploration process even when they know only a little about the data [23].

Numerous visualization techniques have been developed according to the data that has to be explored (one-dimensional, multi-dimensional, textual...) and the interaction and distortion mode that is used to navigate among it. In this context, distortion refers to the fact of altering the properties of the display - while preserving original data structure - in order to better visualize the data. A typical example of that case are the railway station maps, where the distance between stations is not proportional to the real world distance. However, as a benefit of this distortion, occupied space is highly optimized and clearer information is available to the user. A review of some of these information visualization techniques can be found in [44].

The typical nature of the datasets to be explored is multi-dimensional, that is to say, each element is defined by a vector of numbers, usually more than three. In these cases we can not, for example, perform a direct visualization in a 2-dimensional or even 3-dimensional plot. That is the reason why *multidimensional scaling* techniques are commonly used in conjunction with information visualization. Multi-dimensional scaling allows us to map high-dimensional data into a lower-dimensional space while preserving, as much as possible, the original structure of the data. In this way we can project multi-dimensional data points into a 2-dimensional plot taking in account all features that represent the data, and conserving - up to certain extent - the relations between different elements. The idea is that elements that remain close in high-dimensional space, will still remain close in the lower-dimensional one. These techniques are also called *dimensionality reduction*. Probably, the most well-known and widely used (at least in the audio domain) is the *Principal Component Analysis* (PCA), which expresses each one of the dimensions in the latent space as a linear

combination of all the dimensions in the higher space. Other technologies exist such as the *Self-Organizing Maps* (SOM) [24] or the more recent *parametric t-Stochastic Neighbor Embedding* (t-SNE) [44], which learn non-linear mappings between the higher and lower-dimensional spaces (to name a few). Moreover, Self-organizing Maps also perform a *clustering* process by using a reduced set of prototype vectors to represent the data, which makes them particularly suitable for data visualization purposes.

Table 2.3 lists a variety of systems intended to use large sound databases and relates them with the used information visualization techniques.

2.5 Audio Visualization

In contrast with the previous section, we refer specifically to audio visualization techniques when the information we want to visualize is not a set of audio files but only one single file. General information visualization techniques can serve us to generate a contextual view of all the information in a sound database, while audio visualization can be used to focus in one of the items and visually display its properties in detail.

The most straightforward representation is the audio waveform used in the vast majority of audio applications. The problem is that information given to the user is somehow limited. For example, looking at the waveform of a sound file we can have an idea of some dynamic qualities of the audio, but pitch or timbre information are not directly available at first sight. Previously mentioned *Sound Torch* uses simple waveforms plotted in a circular way as the symbols representing each sound file in the scatterplot. Another typical representation is the spectrogram. That seems to be a slightly better approach, but is still difficult to understand even for experienced users (probably because there is too much information). *Rastrograms* [46] provide a novel waveform representation technique inspired in the *raster scanning* method used in most video and image formats. The idea is to display the waveform as a grayscale 2-dimensional image where each pixel corresponds to one audio sample. The intensity of each pixel is mapped from the audio sample with a linear scale where values close to -1 are depicted as darker pixels whereas values close to 1 are brighter. The width of the image is supposed to be *pitch-synchronous* with the waveform, that is to say, the width of the image is determined by the number of samples in each period of the time domain audio signal. As a result, pitch and timbre evolution information are somehow embedded in the representation by observing the way in witch consecutive lines differ from the others. *Rastrograms* are useful for visualizing

System	Visualization	Relevant comments
Timbre Space [43]	3d plot.	PCA is used to reduce the dimensionality of the feature vector to x, y, z coordinates that can be directly mapped in the 3d space. Features extracted include Spectral Centroid/Flux/Rolloff, RMS, LPC and MFCCs.
Sonic Browser [5] †	Scatterplots, Treemaps, Hyperbolic Trees, Touch Graphs and 3d plots.	User can perform PCA to lower the dimensionality of the vectors. Alternatively, it is also possible to select in real time the features to be used in the axis of any of the visualization techniques. Moreover, users can also apply classification and clustering techniques such as K-Nearest Neighbours or K-Means accordingly.
Islands of music [29] †	2d map (see figure 2.2).	Intended for musical genre classification. A variation of the SOM technique is used. It consists in generating three different self organized maps only considering timbre, rhythm or metadata information. These three maps are "aligned" and interpolated in a way that user can gradually change focus from one view to another.
Sound Torch [20] †	Scatterplot.	SOM is used to represent all the sounds in a 2-dimensional plane according to similarity. User is provided with a mouse-controlled *aura* (called the "torch") that triggers the sounds in the databases when it overlaps their position. Sounds are reproduced in a surround environment with the spatialization done according to the display in the plot. In this way user is supposed to quickly find one particular sound in the plot even when he is hearing to a lot of sounds at the same time.

Table 2.3 – Information visualization techniques used in some audio systems.
† Audio features used on these systems have already been listed in section 2.3.

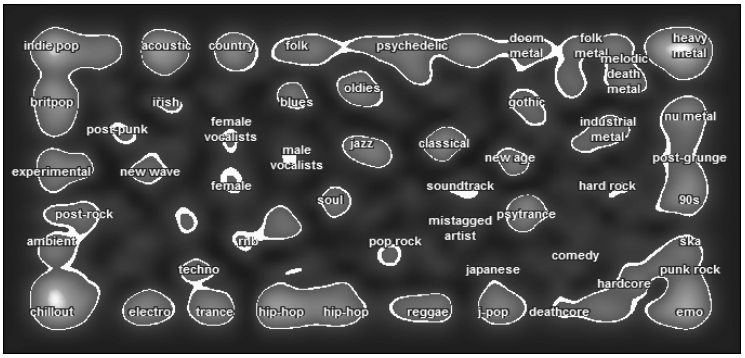

Figure 2.2 – Musical genre visualization in *Islands of music*.

specific type of sounds with simple modulations or timbre structures, but still does not give a general overview of the signal that can be easily interpreted. It seems obvious that we need other approaches that synthesize in some way the properties of the audio and represent them in a more meaningful way. For this purpose, a set of audio features is usually extracted and mapped to some kind of abstract representation that resembles audio properties in a visual form.

In [43], two systems for timbre visualization are proposed: *Timbre Gram* and *Timbre Ball*. Both are based in the extraction of a set of low-level features (the same used for *Timbre Space*, shown in table 2.3) that are later reduced to 3-dimensional vectors using PCA. Sliding windows are used to cut the file in a sequence of frames and 3-dimensional vectors are computed for each one of them. In *Timbre Gram*, a series of vertical color strips are plotted. Each bar corresponds to a frame and its color is a mapping of the vector to either RGB or HSV components. In this way, it is possible to visualize a simple evolution of the timbre for a single audio file. However, the final color that arises from different timbre qualities depends on the particular training set used for the PCA, and it is difficult to make a relation between colors and perceptual understanding of the sound. *Timbre Ball* uses the same information to display a ball in a 3-dimensional space that keeps on moving according to changes in timbre. In that case, 3-dimensional vectors are mapped to x, y and z coordinates, and no color information is used. In contrast with *Timbre Gram*, *Timbre Ball* changes its display in real-time according to the music. Therefore, at first sight we do not have a general overview of the evolution of the timbric qualities of the sound.

Smart Music Kiosk [17], *Conversational Talk* [4] and the technique proposed in BBC research [26], are visualizations intended for navigating through long audio files such as music or recorded radio programs. *Smart Music Kiosk* is an audio player that tries to identify the musical structure of the songs and display it with rectangles corresponding to different parts (*intro*, *verse*, *chorus*...). The idea is that user has a quick overview of the musical structure and can easily navigate through the song. *Conversational Talk* is aimed to represent human conversations by drawing the waveform of the conversation onto a spiral form and paint in different colors the contributions of the different speakers. It works in real-time while the conversation takes place, and users have an idea of who has been talking at any time. For identifying the different speakers, conversation is recorded with different microphones, so MIR techniques are not applied in the process. Finally, BBC article presents a similar visualization to the one proposed in the *Timbre Gram* but intended for long recordings of audio programs. The idea is that three simple features - zero crossing rate, third central moment of the zero crossing rate and 95th percentile of the amplitude spectrum - are extracted for every frame of the signal and mapped to RGB components of consecutive vertical bars. The purpose of the visualization

is to localize certain events in the audio file (like changes in the speaker or musical segments), thus the features analyzed are believed to represent well these changes, but the colors of the representation themselves do not correspond to any specific perceptual descriptions.

Sono Sketch uses a pitch/time grid similar to a spectrogram where users can draw sounds that are later used for querying a database (see figure 2.1). However, this representation can only be created by the user and can not be generated according to features extracted from audio files (see explanation at the end of section 2.3) so, in our opinion, it should be considered rather as a *visual audio synthesis* technique than an audio visualization.

2.6 Search design concepts

In [2], Bates discusses the concept of system involvement in a search process. Modern search interfaces perform automated information retrieval tasks to aid users either in the query definition or in the presentation of results. However, finding the optimum degree of user *vs.* interface involvement in the search is crucial for facilitating the task to the user without constraining its sense of control over the system. The author recommends that search interfaces perform these automated tasks as alternative *recommendations* for the users.

Another relevant concept to be considered is the *dynamic querying* presented in [39]. The idea of dynamic querying is to apply the principles of *direct manipulation* to the database environment. Search results are updated continuously as users type into a box or adjust sliders and buttons to redefine certain aspects of the query. Users seem to react to this approach with enthusiasm, making the search interface both appropriate for novices (which can learn quickly about the behavior of the interface by dynamically observing the impact that redefining queries has on the results) and for experts (that have the ability to define complex queries). A natural consequence of dynamic querying is the concept of *query preview* [41]. As the results of the search are constantly updated, a preview of the amount of results obtained by a specific query can be quickly grasped. The idea is that this preview can help users to avoid getting zero-hit or mega-hit results. In some systems (such as the *Relation Browser ++* [48]), this concept is exploited up to the point of having a dedicated part of the interface for preview purposes that is already navigable, and another one for query specification.

Search tools implementing dynamic queries tend to define them with graphical representations. In [1], Baeza *et al.* review four approaches to graphical query

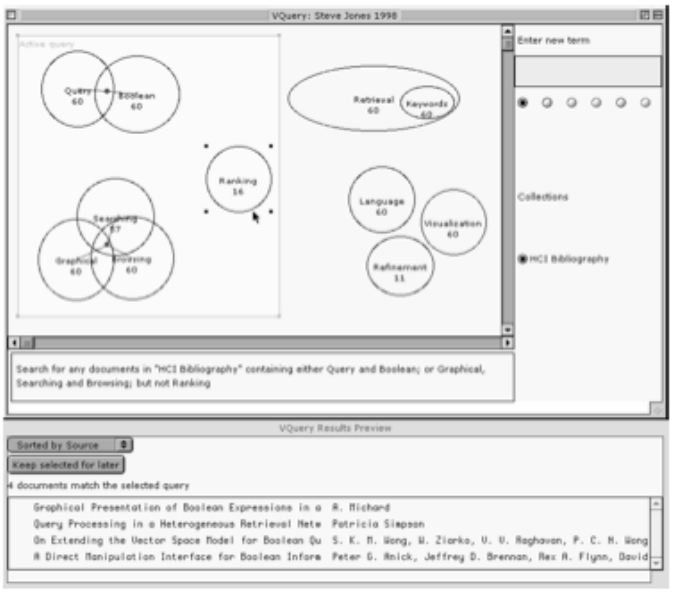

Figure 2.3 – Example of Venn diagram visualization for graphical query specification.

definition. The first one (i), is the use of *Venn diagrams*. That representation consists of a series of ovals with a written word inside that corresponds to a term for the query. By overlapping these ovals, user defines *boolean query* operations such as ANDing different terms. Figure 2.3 shows an example of a Venn diagram for defining a query. Another approach (ii) to define queries is the *filter-flow* model. In this case, query is defined by a sequence of scrollable lists where users select attributes for the query. Each new list represents another constrain applied to the search results feed from the previous list. A waterfall is depicted connecting adjacent attribute lists, and its width is proportional to the amount of items in the database that satisfy all constrains at each stage of the query. The third (iii) method for graphical query specification is the use of block diagrams. Each block represents a term for the query, and can be arranged in a matrix. Blocks in the same row are ANDed, while different columns are ORed. Moreover, they can be activated or deactivated to immediately observe its impact on the search results. Once again, block diagrams are used to mainly represent operators for boolean queries. Finally, another method (iv) named *magical lenses* is explained. The idea is that elements on the dataset are displayed in a 2-dimensional space and query is defined by a virtual lens that can be focused onto a group of elements. This lens represents a specific filter for the query (*i.e.* the presence of a word) and highlights all the items inside the group that satisfy the filter. More than one lens can be placed onto the results space to define complex and parallel queries.

Regarding the presentation of the search results, a commonly used technique is the *clustering*. Similar results among the whole set are grouped together according to a certain measure of similarity. In [47], a system - *Grouper* - which makes use of this concept is presented. That system performs automatic clustering of the results of a textual dataset based on common phrases appearing among them. According to the authors, suitable clustering for search interfaces must follow three general rules: (i) clusters must be coherent in the information they contain, (ii) different clusters must be defined in a way that are efficiently browsable, and (iii) clustering must be done quickly for avoid increasing delay between query and results. In relation with this last point, a common solution is to compute pre-clustering of the data sets before query definition. However, authors point out that after-query clustering is better because different partitions are defined only among the resulting set of documents at hand (instead of all information dataset), and that outcomes more meaningful and coherent clusters.

2.7 Interface evaluation approaches

Evaluation of user interfaces (UIs) is a non trivial task that becomes crucial in the process of designing an interface. Although there are several approaches to solve that problem, there is no general method that can be always applied, and the process often becomes a *soft-science* work around. Back in the eighties, Card, Moran and Newell stated that UIs evaluation needed of more research and accurate techniques in order to be able to design interfaces fitting human nature in terms of *human sensory limitations* (what humans are able to sense from different communication channels with an interface), *human knowledge* (the ability to interpret, predict and understand the behavior of a system) and *human environment* (the tasks the system is used for in a social context) [10]. Attending to that claim, they suggested the *Model Human Processor*, introduced as a "model for making engineering calculations of human performance" (see figure 2.4). Since then, other models such as *Goals, Operators, Methods and Selection rules* (GOMS), *Keystroke-Level Modeling* (KLM) and other derivates like *Cognitive Complexity Theory* (CCT), made their appearance with the same intention of giving some objective guidance to user interface design processes [28]. The idea behind all these models is to predict human performance - in terms of required *time* for actions such as a key-stroke or mouse pointing - during the interaction process with the interface, and focus the design in optimizing the results according to this model, even without the need of *mookups* or prototypes.

Nowadays, higher-level approaches for the evaluation process are used. In [21], a comparison between four commonly used different methods - *heuristic evaluation*,

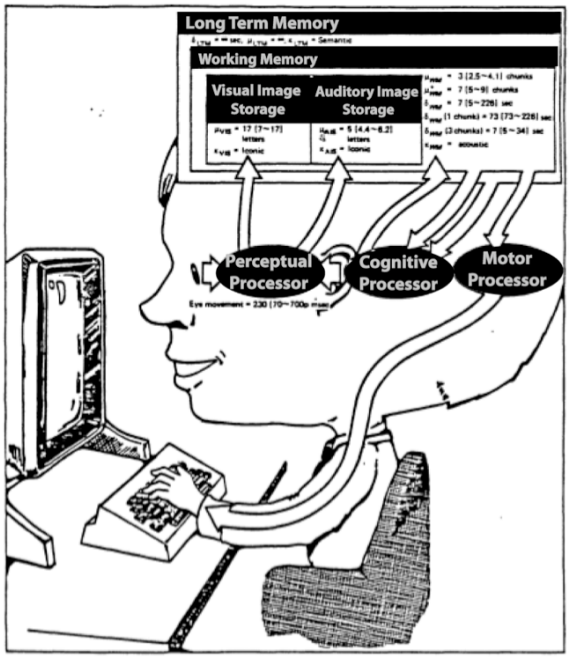

Figure 2.4 – Diagram of the Model Human Processor as defined by Card, Moran and Newell. The interaction with the interface is simulated at a very low level using this model, which defines "a set of processes, memories and their interconnections together with a set of principles of operation" to *calculate* user responses.

user testing, *established guidelines* and *cognitive walkthrough* - is presented, with usability testing being the most popular[1]. Through heuristic evaluation, interfaces are examined by UI experts that identify problems and recommend modifications for improvement. According to the authors, this method can lead to the identification of many and severe problems, but UI experts are not always available or hard to find, and this is an important drawback.

In the other hand, usability testing consists in proposing a variety of tasks to a group of users and see how they respond. Both quantitative and qualitative data (*i.e.* computing the time needed to accomplish a task or delivering a questionnaire respectively) is collected and analyzed to understand the problems and possible improvements that could be applied. The definition of the tasks to accomplish is crucial to obtain reliable evaluation, therefore, expertise is needed in the test design step. In [38] we can find some guidelines for the preparation of those tests. In general, this method performs good in identifying important problems in the interfaces, but

[1]Raw estimation according to the number of results obtained by querying *Google Scholar* with the name of the four methods. All methods obtain less than half a million results except for user testing, which returns more than three million and a half (accessed 31/03/2010).

it needs a certain amount of users for the test, UI expertise in the design step, and a lot of time to perform the whole evaluation. This means that, if done properly, the cost of the evaluation process can increase quite a bit.

The two remaining methods are aimed to solve the previously commented drawbacks, specially the need of UI experts and users to test. Using established guidelines, interfaces are evaluated by the developers themselves by checking that it satisfies a set of general rules that define *how* interfaces should behave. The obvious problem of that method is that any aspect which is not considered in the guidelines will not be checked in the testing, thus some important issues could be missed. Finally, cognitive walkthrough proposes a variation to the usability testing that is again performed by the evaluators themselves. The problem of that last method remains in the lack of methodology for the task definition (expertise), but still can give insight on the problems that can arise of an interface.

Chapter 3

Proposed system

3.1 Software Environment

For the implementation of the prototype we have used a combination of several technologies that work in combination. The general structure is based on a *client-server* scheme, although for our prototype they both run locally. In general terms, the server side deals with the audio database and is in charge of performing the queries, returning the results and calculating their organization in the map. The client part implements the graphical interface that allow users to define queries and display the results according to the information returned by the server. Moreover, the client is also in charge of downloading and reproducing the sounds that are returned from a query. Figure 3.1 shows the general block diagram of the system.

Server side

The server part is implemented with a *Python* script that communicates with the interface through *Xml-Rpc* protocol. Python programming language was chosen for its simplicity and the possibility to access *Essentia* and *Gaia* libraries (both developed in the *Music Technology Group*, *Universitat Pompeu Fabra*).

Essentia is used to analyze all audio files that conform the database and extract relevant features to describe each sound. This process is done *off-line* and only once. Extracted information is turned into a Gaia database that is later used to perform the queries and some other additional operations. More information about these aspects can be found in the following sections.

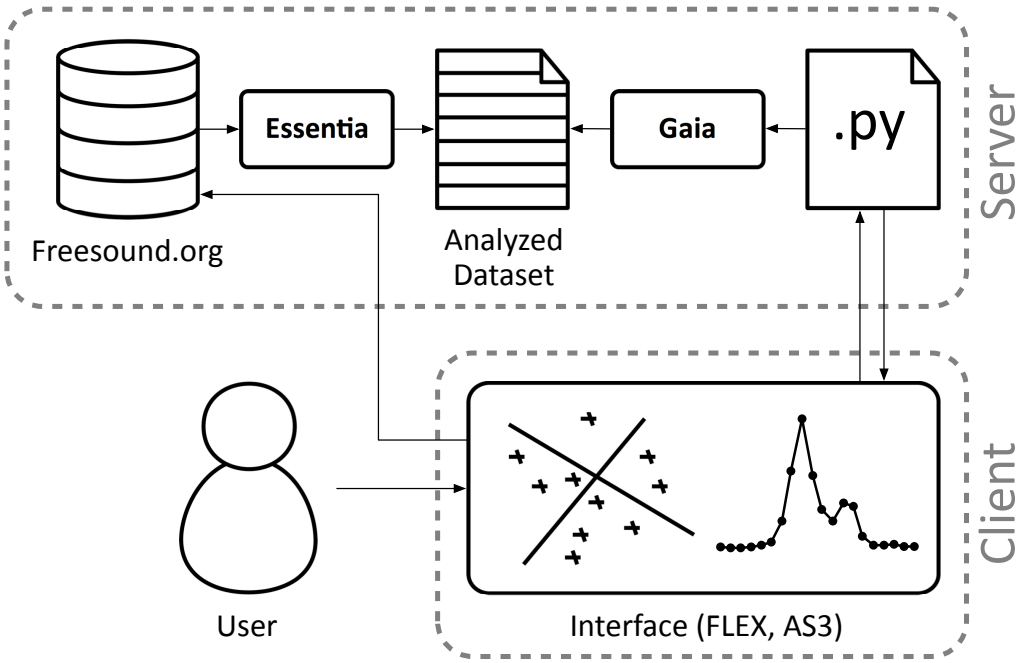

Figure 3.1 – Block diagram of the proposed system.

Client side (interface)

The graphical interface has been implemented using *Flex*, a software development kit released by *Adobe* and aimed to create cross-platform rich internet applications based on *Adobe Flash*. The main reason for choosing *Flex* as the framework for the implementation of the graphical interface was that it directly runs in any web browser (with Flash plug-in installed) and provides many facilities aimed for internet applications (such as the streaming of online audio content with just a few functions). Moreover, it also provides simple functions to create controls and nice graphic objects that makes it easy to quickly program some of the most typical aspects of an interface (such as buttons or windows).

Sound database

As the source of sounds for the database we are using all sound files stored in *Freesound.org* (another project of the Music Technology Group). Freesound is a huge online sound database (around 90.000 files at the time this report was written) that has grown a lot during the last years. All the sounds stored in the database are shared under a *Creative Commons* license and can be accessed from a public web interface through a text-based search engine. Nowadays, Freesound has a big community of users that are constantly uploading and reusing the files stored in it.

As users themselves are the ones that freely upload sounds in the database, the information we can find is completely heterogeneous. We can find lots of sounds from sampled instruments, recorded melodies or drum loops to voice recordings, sound effects, abstract synthesized sounds or environmental recordings (just to name a few categories). From our point of view, this is interesting because everything can be used creatively in a musical context, but it also makes hard to think of a way to describe any of these sounds in a simple graphical query definition approach.

3.2 Query design

By *query design* we mean *how do users introduce* the information in the interface and *which kind of information* they have to introduce. Regarding the first point, we already explained that we are taking a graphical approach that differs the traditional text-based query specification. About which kind of information to introduce (and this also constrains how it should be introduced), we take the consideration that our search interface is aimed to be useful in a musical context, thus we give a special emphasis on the musical properties of the sounds in the database. Moreover, the second consideration commented in the introduction is that we are focusing on the perceptual aspects of sounds rather than on the source that causes them. According to these reasons, we define four different perceptual aspects which we think are relevant from a musical point of view, and choose sound descriptors that can be representative of them. What follows is a brief description of which are those perceptual aspects and why we choose them, later we give a more detailed explanation showing the used descriptors and how they are graphically introduced in the interface.

- **Envelope:** by envelope we mean the temporal evolution of the energy in the sound. Defining the envelope we can differentiate from short or long sounds, with slow or fast attack and decay (among others). In a musical context, those characteristics are quite relevant in determining the *function* that a sound carries on in a production (long sounds for background atmospheres, short sounds with fast attack to build rhythmic patterns, sustained or unsustained notes for a melody...).

- **Timbre:** this is often used as a word regarding any sound property that is not pitch, loudness or duration [40]. In our case, we specifically refer to the spectral content of sound from a static point of view (without considering evolution or variation), that is to say, the different range of frequencies of sound that tend

to have more energy. In our opinion, this is interesting because we can define the general *color* of the sounds we are looking for.

- **Pitch:** in this case the concept is quite obvious. If we want to find sounds that correspond to a specific note it is useful to define a pitch on the query.
- **Tonal content:** finally, and from a strictly musical point of view, we think it is also interesting to query sounds for they tonal information. That can allow us to search for chords or melodies in a specific key.

3.2.1 Used descriptors

Regarding envelope

When thinking of a way to characterize the envelope of a sound we come upon two different approaches. In one hand, we can use a model based on Attack/Decay-/Sustain/Release (ADSR, as in most synthesizers) using descriptors such as *log-attack time*, *effective duration* or *strong decay*. In the other hand, we can compute the envelope by analyzing frame by frame the audio file and extract the energy. ADSR model seems simpler to introduce from the point of view of the user, but our concern was that not all sounds in our database can fit this model. For the sounds that represent *single events* such as single notes, percussive elements or some sound effects, ADSR could work fine, but we can also find other things like recorded melodies or drum loops that are hardly characterized by a model like that. Therefore, we thought that extracting the energy frame by frame could be a better approach. In fact, in our prototype, envelope is the only perceptual characteristic where we consider its evolution across time. All other aspects are considered as *global* (invariant) descriptors. This specific point is discussed on section 3.2.2.

In our prototype, users are able to introduce the envelope specification by drawing a line in an *amplitude/time* grid. The time is limited to ten seconds (this point is discussed on section 3.2.2) , and the amplitude is logarithmically mapped to the vertical axis of the grid. We are using a logarithmic scale because it is closely related to human perception of loudness. By using a linear scale, we see that making queries with different constant values on the amplitude axis does not give relevant results (*i.e.* looking for an amplitude either closer to the maximum or the minimum give results with similar perceptual loudness). However, using the logarithmic scale we observe that results go in better accordance with the query specification. Figure 3.2 shows the grid we are using and a screenshot of the envelope definition in the prototype.

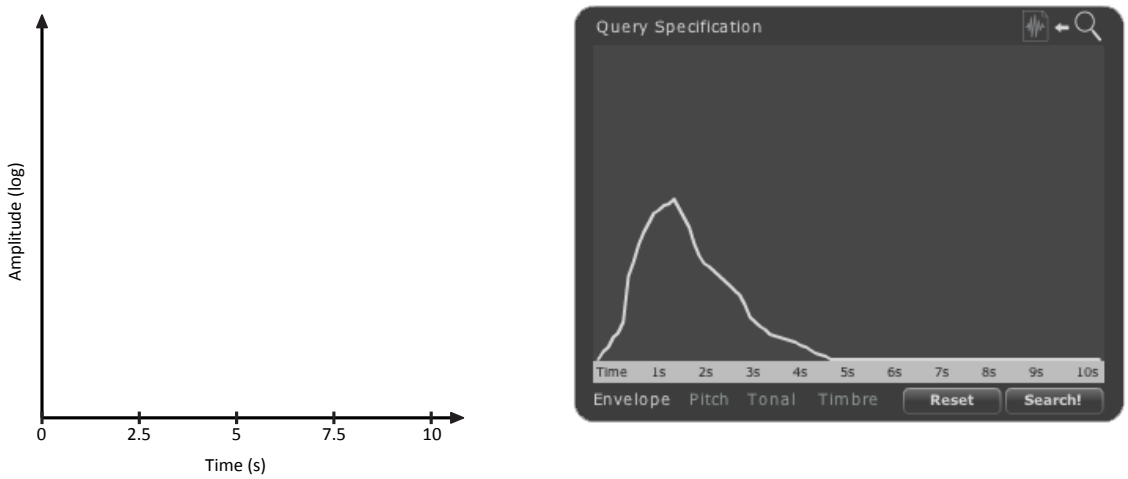

Figure 3.2 – Axis for the envelope specification.

Regarding timbre

To characterize timbre we are using a rather simple approach which consists on establishing a series of frequency ranges (bands) and computing their energy. These frequency bands are set simulating human auditory system behavior (perceptual model). Probably the most popular approximation used in this kind of analysis is the Mel scale, which is linear in the low frequency range (up to 100 Hz) and logarithmic above that point. However, in our prototype we have used a Bark scale, which "can model a better approximation of the human auditory system" [30]. Frequency range from 0 Hz to 27 kHz is divided into 27 bands[1] that correspond to the critical bands of human auditory system.

In our query specification, users can introduce timbre information by setting the amplitude of each one of the 27 Bark bands. Each band is represented by a point in an *amplitude/bark scale* grid and connected with the adjacent bands with a line (to give the sensation of continuity in the spectral energy distribution). As has been done with the envelope, amplitude axis also uses a logarithmic scale. Figure 3.3 shows the grid we are using and a screenshot of the timbre definition in the prototype.

[1]Normal definition of Bark bands establishes 24 bands from 0 to 15.5 kHz. However, in the Essentia implementation, the range is extended up to 27 kHz. Furthermore, first two bands (0 Hz to 100 Hz and 100 Hz to 200 Hz) are split into half for better resolution, leaving a total of 27 bands

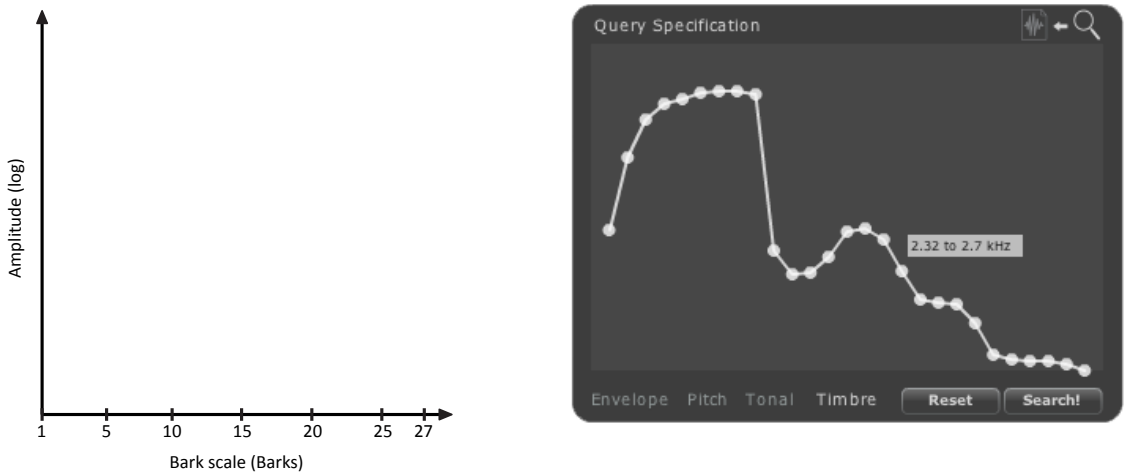

Figure 3.3 – Axis for the timbre specification.
When users edit timbre, a label appears showing the range (in Hz) of the currently edited band.

Regarding pitch

Pitch is characterized using the output of the pitch detection algorithm implemented in Essentia. This method is a variation of the YIN algorithm, performed in the spectral domain [7]. As an additional aspect for describing the pitch we are also using a measure of *dissonance*. Dissonance is extracted with Essentia and based on the roughness of the spectral peaks. The smoother the spectral envelope, the less dissonant the sound is considered. With this measure we add another dimension that is useful to set the "purity" of the pitch (it also acts as a measure of "pitchness").

To introduce this information, user is given with a *dissonance/frequency* grid where a single point can be defined, thus specifying both dimensions. Frequency axis uses a logarithmic scale to better represent human perception of pitch (semitones are equally spaced among the axis), and ranges from E1 (41 Hz) to B7 (3951 Hz). The range of this axis is defined to represent the 95% of the data (see section 3.5.2). Figure 3.4 shows the grid we are using and a screenshot of the pitch definition in the prototype.

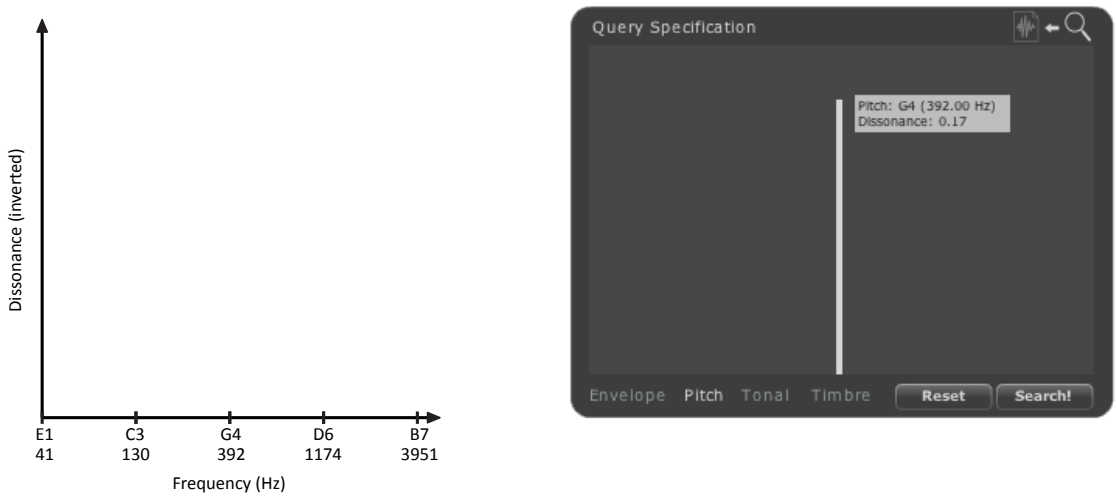

Figure 3.4 – Axis for the pitch specification.
When users edit pitch, a label appears showing the current pitch value (both the musical note and the frequency in Hz) and the dissonance value.

Regarding tonal content

To represent the tonal content of an audio file we compute the Harmonic Pitch Class Profile (HPCP), which is a twelve-dimensional[2] vector where each component represents the intensity of a semitone (on a chromatic scale) regardless of the octave. To compute the intensity of each semitone, the algorithm finds all the spectral peaks that fall in the frequency range corresponding to the same note in the different octaves. HPCPs are also usually called *chroma* features. As an additional dimension used in the query process for tonal content we use the *Key strength* extracted by Essentia. This value gives a measure of how prominent is the key in a sound file. Users do not directly specify this value but it is imposed as a constrain when performing queries using tonal content (see section 3.5.2). It is useful as a way to ensure that tonal content of returned results is meaningful.

In our prototype we represent chroma features with twelve bars whose height can be set independently in an *amplitude/semitone* grid. In this way, users can define the relative importance of each semitone compared to the others (*i.e.* if a user wants to search sounds in C Major, he can raise the bars corresponding to C, E and G). First bar corresponds to C whereas last bar corresponds to B, and are painted in

[2]Actually, HPCPs can also be computed using subdivisions for each semitone, therefore the number of resulting bands is $k*N$, where $k = 1, 2, 3...$ In our analysis we are using three subdivisions for each semitone, thus we have a total of 36 bands that later are averaged to form the final 12 bands representing the intensity of each semitone.

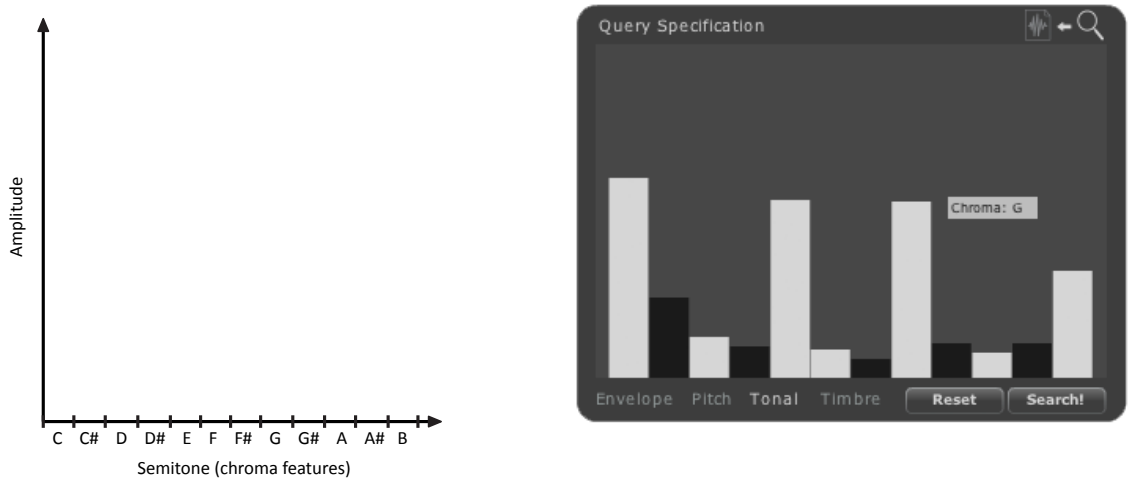

Figure 3.5 – Axis for the tonal specification.
When users edit tonal content, a label appears showing the semitone corresponding to the currently edited bar.

black or white resembling the colors of a piano keyboard. Figure 3.5 shows the grid we are using and a screenshot of the tonal content definition in the prototype.

3.2.2 Reliability of the descriptors

As we already commented, leaving aside the envelope characterization, we are not considering the temporal evolution of the descriptors in any case. To extract a global value after a frame by frame analysis of the audio file we compute the mean across all frames. This means that our descriptions represent an average of all the variation that is found inside the analysis. This decision was taken by means of simplification, not only from the point of view of the implementation but also for designing how users specify the query (specially in the case of multi-dimensional features). As a drawback, we have to be concerned that the more variation there is inside a file, the less reliable its description gets. In an effort of minimizing that effect, we applied a time limitation to all the database, only considering the subset of files that last for less than ten seconds. With this limitation, we are assuming that shorter sounds have less variation, therefore we are "partially" solving that issue. Obviously, ten seconds is still a long enough time period to allow important variations, but at least we are getting rid of some files that would presumably interfere our queries.

3.2.3 Query interaction

Up to that point we have said that query specification can be done by setting four perceptual aspects and we have explained how do users introduce the corresponding information in each case. Another important aspect of the query design is that not all perceptual aspects have to be filled in order to perform a query. As an example, our approach is that users might be interested in searching *bright* sounds regardless of their other qualities, or might want to find sounds that fit a specific tonality no matter the duration or the timbre. For this reason, in our prototype each one of the perceptual aspects can be independently activated or deactivated. Furthermore, we are also aware that those perceptual aspects are not completely orthogonal (specially tonal content, pitch and timbre, which are all derived from the spectral content), and that using them all together may interfere each other and produce confusing results (typically if there are no sounds in the database that closely match target descriptors). This is another reason to think that it is useful to allow queries without defining all the descriptors.

Moreover, users can specify the query information using two different approaches. In the first one, they can start from scratch and define any of the perceptual aspects to set a target. However, sometimes this might be difficult, particularly in the cases when we do not know exactly how to specify the target using the given tools (*i.e.* we want to define a *dull* timbre but we do not know which frequency bands should be particularly reinforced). In those cases it might help to use the second approach, which consists in using any result to *query-by-example*. We can roughly define what we want to find, explore preliminary results and then refine the query by taking one of them as an example. Again, when querying by example we do not need to use all perceptual aspects at once, we can select the aspects of a sound that we are interested in. Therefore we can perform queries such as obtaining sounds with the same pitch as a given one, or with the same envelope.

3.3 Search Engine Results Page design

In a typical search engine such as *Google*, Search Engine Results Page (SERP) is the listing of the results that are found in response to a query. Web pages are listed according to some criteria of relevance, and a number of pages with N results each are generated and can be navigated. Although the interface we have designed differs a bit from this concept because we are not presenting the results with different pages, we still call the design of *how* results are presented to users as SERP design.

In our prototype, search results are represented as points in a two-dimensional map. In each query, a new map is build according to all returned results. The number of returned results can be set by the user. For projecting the points in the map, we have used two different approaches: direct mapping of the descriptors and dimensionality reduction techniques. The following sections describe these approaches and discuss other topics regarding SERP.

3.3.1 Direct mapping

The simpler approach for displaying search results in a two-dimensional space is the mapping of the descriptors to the horizontal and vertical axis. The idea is that each one of the perceptual aspects commented in section 3.2.1 can be used to define one of the axis of the map. We could make an analogy with traditional SERPs by understanding that we are *sorting* each axis according to one of the perceptual aspects that define sound. An obvious issue with this approach is how to map multi-dimensional vectors (*i.e.* timbre information given by Bark bands) into a single value of the axis. To overcome this issue, we did not use the same descriptors for the query definition but choose others that were representative (and probably easier to understand) of each perceptual aspect. Table 3.1 shows the relation of the descriptors used in the query definition and in the results representation. What follows is a brief description of them.

- **Duration:** users are able to sort the results in one of the axis according to their duration. Using this organization, short sounds are projected in the left part of the map (or at the bottom if duration is set on the vertical axis) while long sounds fall in the opposite side. This is useful for quickly locate the region of interest among all results according to their length.
- **Spectral centroid:** for organizing the sounds according to their timbre we use the spectral centroid feature. This feature resembles the *brightness* of a sound, and projects them according to their energy concentration in the spectral domain. Sounds with low frequency content are projected in the left side of the map (or at the bottom) while sounds with high frequency content (bright sounds) fall in the opposite side.
- **Pitch:** this mapping is much more simpler as pitch itself is a single value. Using this sorting mode, sounds with lower pitch are projected in the left side of the map (or at the bottom) while sounds with higher pitch fall in the opposite side.

Perceptual aspect	Query definition	Results representation
Envelope	Temporal energy	Duration
Timbre	Bark bands	Spectral centroid (*brightness*)
Pitch	Pitch, dissonance	Pitch
Tonal content	HPCPs	Key

Table 3.1 – Used descriptors in the direct mapping approach for the results representation (in relation to the query definition).

- **Key:** finally, for the case of the tonal content, we are using the Key extractor from Essentia (based on [16] and [42]). This algorithm returns the best matching key estimate for a given HPCP (one of the twelve semitones plus the scale, either major or minor). We use key information regardless of the scale type to project the sounds in the map, therefore, this organization is discrete and has twelve possible positions. Sounds are organized from left to right (or bottom to top) representing keys from A to G#. Some issues arise when the results of a query do not contain sounds with a prominent key. In these cases, that sorting mode does not help to orientate on the map. However, it can be useful to discriminate among the results when looking for sounds with a specific key.

Users can decide which descriptor to map in each axis by using a menu at the top of the interface. To create this mapping we use normalized versions of the descriptors. This ensures that almost all possible values fall in the range from 0 to 1 (therefore, the projection in the map is somehow "controlled"). More details on the normalization are given in section 3.5.1.

3.3.2 Dimensionality reduction

The second approach to project query results on the map is the use of dimensionality reduction techniques to transform a feature space formed by all the descriptors used in the query definition into a two-dimensional xy coordinate system. By using this approach we expect that sounds with similar perceptual characteristics get projected close in the map. That is to say, the map is organized in a way that closer points represent similar sounds, thus, some clusters may appear to give an overview of the kind of results returned by the query. We understand this feature as an "automatic" sorting of the results.

In our prototype we are using Principal Component Analysis (PCA) to perform the dimensionality reduction. PCA is a simple but widely used technique that decomposes a feature space into the N principal components that better represent it. These principal components are expressed as linear combinations of the features in the high-dimensional space that account for as much variability in the original data as possible. In that way, a N-dimensional space is created preserving as much as possible the structure of the original data. In our case, we are performing a drastic reduction of a feature space formed by all the descriptors included in the query definition (see section 3.2.1) into two dimensions used as coordinates for the map. At the time of each query, a new map is computed taking in account all returned results.

For our application, what is important from a dimensionality reduction technique is that the distances between the points in the high-dimensional space are still preserved in the resulting map. That is to say, that closer points in the original space still remain close in the reduced version. We have observed that PCA does not fit that requirement very satisfactorily. Figure 3.6 shows an example of the problems we find with PCA in the lower-dimensional space. There are other techniques such as *Multidimensional Scaling* (MDS and its variations) that are specially focused on the preservation of those distance relations between spaces, and also other techniques that learn non-linear mappings of the features of the high-dimensional such as *Self-Organizing Maps* (SOM), that could probably better fit in our problem and have been used in other music-related applications [13, 20]. However, the implementation of those techniques in our prototype would have required a lot of time (algorithms should have been integrated into Gaia) and would have supposed much more computation time at the creation of each map (thus, making the response to a query slower). Moreover, PCA was already implemented in Gaia and that made our work much more easy.

We also thought in the possibility of using pre-computed maps (we call them static maps) to be able to run those other algorithms without the need of implementing them on Gaia (in a platform like Matlab or Python) and avoiding the issue of making query responses slower. However, building a map for that big quantity of files in the database (around 60.000, see section 3.5.1) was computationally very expensive and required such a big quantity of memory that could not be done with a normal computer (we estimated 14 GB of RAM just for the first step of MDS computation). We are aware that the use of static maps could have been exploited, but we decided to leave it for future work as the concept of the interface changed a bit with this approach.

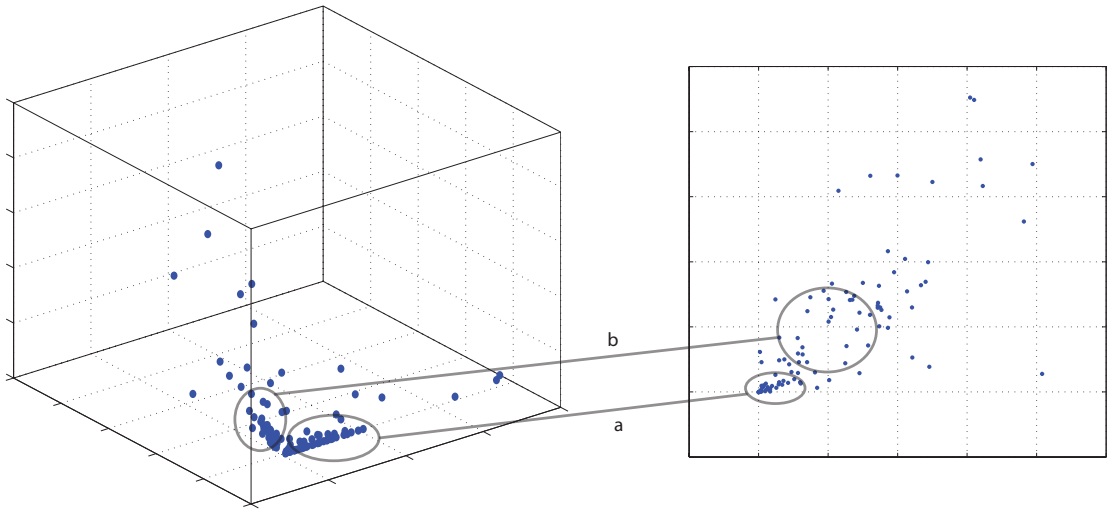

Figure 3.6 – Distance preservation on the PCA projection.
In this figure we can see that the relation between points in the higher-dimensional space is not always preserved after performing PCA. In 'a', we can see a cluster of points that is still conserved in the two-dimensional map, whereas in 'b' we can see another cluster that is spread after the transformation.

3.3.3 SERP interaction

Users can navigate the results with typical actions such as *zooming* in and out and *dragging* the map. This is specially useful when a number of points are projected really close and overlapp themselves. Moreover, zooming out can help in giving an overview of the different clusters of sounds that are created in the projection.

Regarding the exploration of particular results, users are able to select any of them by clicking over the corresponding circle, and watch the visual representation of their perceptual aspects (the descriptors) in the same window where the query is performed (see section 3.4). This allows to make comparisons between the target that has been set and the obtained results and, as has already been commented in section 3.2.3, it gives the opportunity to use perceptual aspects of one result as a target for a query-by-example. Furthermore, some of this information contained in each result is directly available to the user without the need of selecting any result. This is done using a simple mapping that represents the duration of a sound with the size of the displayed point in the map (the bigger the longer) and the spectral centroid with its color. Spectral centroid (normalized from 0 to 1) is proportionally mapped to the amount of red and inversely proportional to the amount of blue (in a RGB coordinate space, green is set to 0). In this manner, sounds with predominant

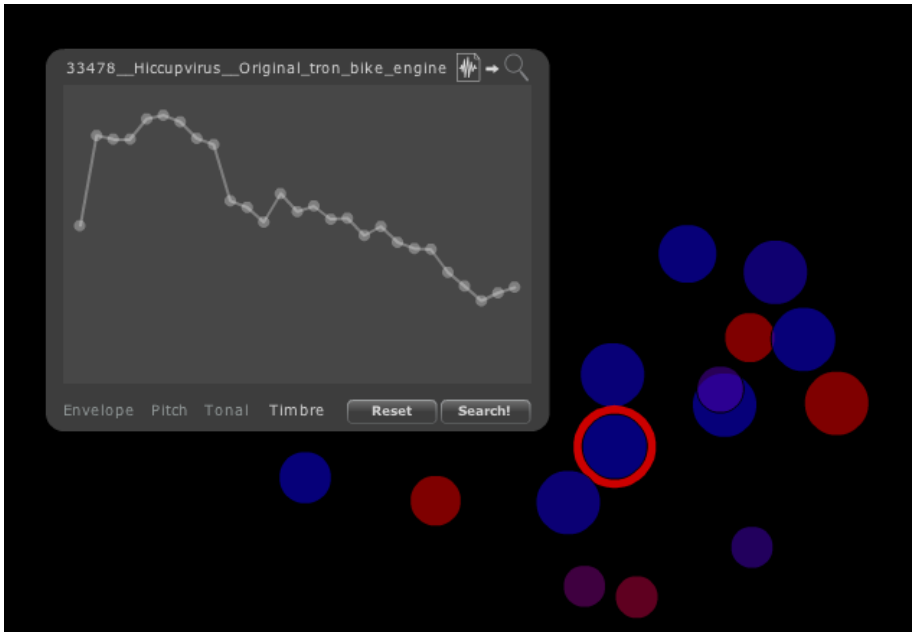

Figure 3.7 – Screenshot of the results representation.

low frequency content are *bluer* while sounds with high frequency concentration tend to be more red. Figure 3.7 shows a screenshot of the prototype where these simple mappings and the graphical representation of the perceptual aspects of a result can be observed.

Another way to explore the results given by a query is by listening to them. By moving the mouse over any result in the map while pressing the *Alt* key, sounds start reproducing (pressing *Ctrl* key they stop). This action is done on-line, and the audio files are reproduced in streaming directly from Freesound.org. As all the sounds are quite short (less than 10 seconds) and are compressed in mp3 format, the loading time is really fast and there are no important delays between the triggering of the reproduction and the reproduction itself. However, if several sounds are triggered at similar times, their reproduction overlaps creating a soundscape. This can make hard to identify which sound belongs to which point in the map. To aid in this issue, a circular progress bar appears around the sounds that are being reproduced, and their transparency (of the points) changes according to the amplitude during reproduction time. Furthermore, points that are located on the left side of the screen are being progressively panned to the left channel of the speakers (the more distance from the center to the left, the stronger the panning and vice-versa). Consequently, sounds panned in the center are the ones located at the center of the screen, regardless of the zoom or the position in the map. With the use of these cues we thing that localizing the sounds that are being reproduced becomes an easier

task. Nevertheless, soundscapes that are created by reproducing lots of sounds at the same time can be quite interesting.

3.4 Interface and interaction design

We already explained the approach we have taken for the query definition and the results representation. In this section we will briefly present how those concepts are merged together in the interface we have designed.

Inspired by the already cited statement by Morville and Callender [27] "search goal is more than finding, search should become a conversation process where answers change the questions" we designed an interface aiming to reinforce a continuous *conversation* between query and results. Our idea is that by exploring the results user has more information to refine the query, and this is done continuously until desired sounds are found. Each time a new query is defined, a new map is generated. Smaller changes in the query definition provoke smaller changes in the results map. Following this idea, we think that query specification and results representation should occupy the "same" space on the interface, they are thought to be used together.

Figure 3.8 shows an scheme of the different elements found in the interface and how they are organized. As we can see, most of the interface is used to display the map of results (number 1 in the figure). A small menu bar is located at the top part of the screen and gives some minimal controls to *open and close the query panel* (where queries will be specified), set the *number of desired results*, choose the *sorting mode* for the map and finally *stop all sounds* that are being reproduced (in the figure, numbers 2, 3, 4 and 5 respectively). At the bottom part, there is another little space which is only used to provide some information to the user regarding the current processes that are being performed (such as indicating when a query is being performed or if there are any communication problems with the server, number 6 in the figure).

The query panel (number 7 in the figure) is formed by a floating window (draggable) with a few controls. First, there are the *reset* and *search* buttons that are used to set the query specification to its default state (number 8 in the figure) or to actually perform the query with the introduced descriptors (number 9 in the figure). At the left side, we can see the names of the four perceptual aspects of sound that can be defined and explored. Those labels work as tabs and only one can be selected at a time. When a label is selected, the information projected in the upper box corresponds to the current perceptual aspect (number 10 in the figure). We

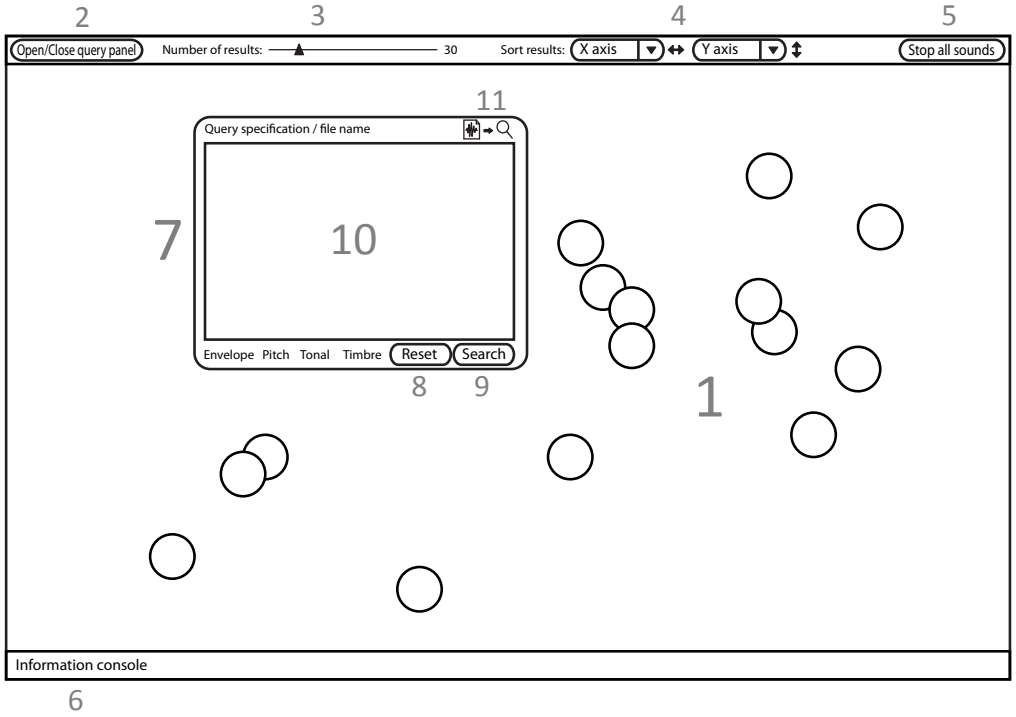

Figure 3.8 – Scheme of the interface layout.

have already shown in section 3.2.1 how each perceptual aspect can be introduced for the query. The title of the window changes from "Query Specification" to the name of a given result when we are exploring its representation instead of displaying the current target for the query. We can toggle between what we call the *file mode* (displaying single result characteristics) and the *query mode* (displaying target descriptors) by clicking the icon in the top right corner of the query panel (number 11 in the figure).

Individual aspects of the query definition can be enabled or disabled (used or ignored in the search) by *Alt*+clicking in the box. Opacity of the displayed descriptor is reduced when it is deactivated. Query-by-example can also be done in a similar way. When we are exploring the contents of a given result, all descriptors are displayed as deactivated. We can activate them using the same procedure (*Alt*+click) and go back to the query specification having copied that particular descriptor. If we want to query-by-example using all four perceptual aspects we can also directly click the search button while we are in file mode (the name of the file is on top of the query panel).

We expect that allowing users to explore the visual representations of the perceptual aspects of the results in the same way as the query is defined, can made easier to understand the meaning of the descriptors and, at the end, of the query

definition. In some way, this working mode wants to provoke a *learning by example* kind of interaction that may help users to autonomously learn how to define good queries. Although targets for the query are defined using the concept of sound descriptors - and this might be a bit difficult to understand for someone that is not familiarized (how to translate the idea of a sound into its descriptors) - we think that this kind of interaction can help to quickly learn that concepts in an intuitive way (*i.e.* making relations with the resulting shapes of the Bark bands and the timbre of obtained results, without the need ok knowing what a Bark band is). In our opinion this is supported by Keim [23], "visual data exploration is known to be particularly useful when little is known about the data".

3.5 General implementation aspects

In this section we explain some important aspects regarding the implementation of the prototype. Our intention is not to provide specific details on the implementation but rather give an overview of the different processes that are performed in relation to the design of the interface. We start by describing the process of building the database with Gaia, continue with the description of the Python module that acts as the server and end explaining some aspects of the Flex application.

3.5.1 Building the database

As has already been said, our database is built with the contents of Freesound. Figure 3.9 shows the different steps that have been followed for that purpose. What follows is a description of these steps.

1. **Analysis:** we performed an analysis using Essentia to extract the features commented on sections 3.2.1 and 3.3.1 among around 60.000 sound files. As

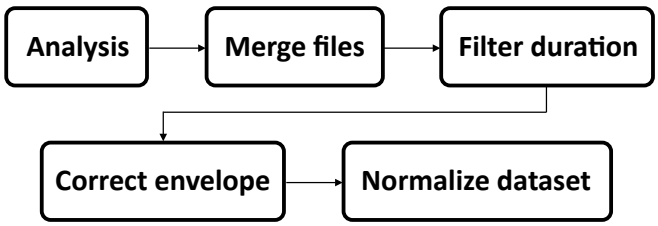

Figure 3.9 – Steps on building the dataset.

output, we obtained the same number of *.sig* files containing the metadata for each sound.

2. **Merge files:** after performing the analysis, all *.sig* files are loaded into Gaia and merged into a database containing all the information in a single file. This process takes a long time but only has to be done once. The result is a *.db* file that contains the information from all audio files and drastically reduces the size.

3. **Filter duration:** the next step is to apply the time limitation explained in section 3.2.2. For this purpose we filter out all files that last for longer than 10 seconds. In this manner, we reduce the size of the database down to around 40.000 files. To apply this filter, we use the *duration* descriptor extracted by Essentia. We are aware that there are some sounds with silence at the beginning or the end, and that our descriptor considers the full length of the file. That means that in some cases the annotated duration might not be completely representative of the sound. However, in any case sounds will last longer than 10 seconds.

4. **Correct envelope:** for the specific case of the envelope information, Essentia extractor computes an energy value at each frame analysis. This means that our envelope descriptor will have a different length depending of the duration of the sound file (variable-length descriptor). For our purpose this is a problem because we can not directly compare the envelope of different sounds if its length is variable. To avoid that issue, we transform this variable-length descriptor into a fixed-length descriptor. As we know that the longest sound will last a maximum of 10 seconds, we define a vector of 100 points that will correspond the energy of a 10 seconds file. That is to say, we want to have the energy value of each audio file every 100 ms. In our analysis, hop size is set to 1024, thus, for a sample rate of 44.1 kHz, energy values are calculated every 23.22 ms. We correct the envelope by assigning at each position the original energy value corresponding to the closest moment in time (to the current multiple of 100 ms). Remaining positions in the 100 point vector are filled with zeros.

5. **Normalize dataset:** the last step in the process is to apply a normalization to the dataset. In this manner, we make sure that the values from all descriptors are inside a similar range. This is useful both when performing the queries and building the maps. In the case of the queries, once a target is set, Gaia performs a similarity search among all the database, and returns the N most similar results. By using a normalized dataset, we make sure that all

the descriptors account with a similar relevance in this distance calculation. The same concept can be applied to the PCA computation for the maps. To perform that normalization, Gaia already provides a built-in function. Normalization is done to output values in a range from 0 to 1, however, to avoid the influence of outliers with extreme values (that could "distort" the output), we discard 5% of the data (the 5% that is more distant from the mean) and perform the normalization without considering them. In this way, all the data is normalized and has a range from 0 to 1 without being affected by that 5% that could be considered outliers. Outliers are also normalized but can take values above 1 and below 0 (the less far from the mean they are, the closer to this range they will be).

3.5.2 Python module

The Python module implements all the functionalities of the server. It is the part of the implementation that can access our Gaia dataset. Figure 3.10 shows the block diagram of the functionalities of the server. As we can see, it basically loads the dataset, initializes the Xml-Rpc server and remains in a loop waiting for petitions coming from the interface. There are two offered services which are "Get features" and "Perform query". What follows is a brief description of these services.

- **Get features:** that service returns a list of the features of a specific file on the database. It is used by the interface when a particular result is selected and it has to display its perceptual aspects in the query panel. The interface asks for the features using the filename.
- **Perform query:** when user defines a target for a query an press the "search" button, the interface requests this service and passes the target features set by the user. Figure 3.11 shows the different process performed by the server when this service is requested.

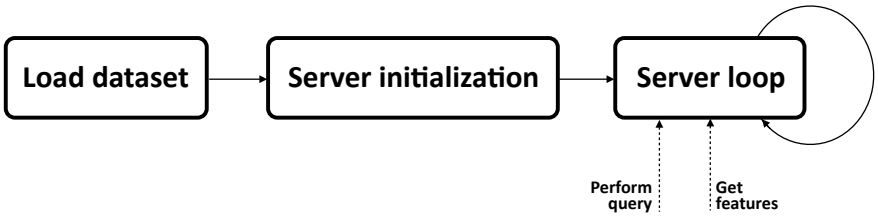

Figure 3.10 – Block diagram of the Python module.

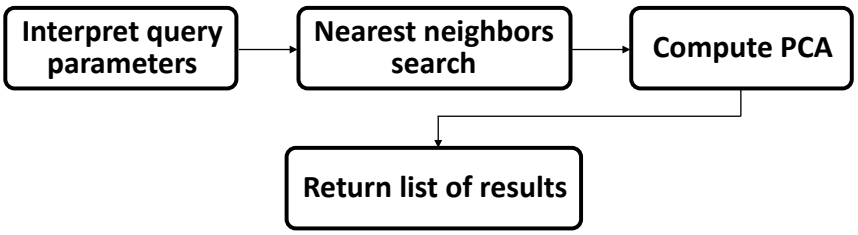

Figure 3.11 – Steps on performing the query.

First of all, query parameters are interpreted and a new point for the database is built using the features given by the user. As we have said, database is normalized in a way that 95% of the descriptors have values inside the range form 0 to 1. The amplitude axis used in the query specification are also normalized from 0 to 1. In this manner, we are automatically setting the maximum and minimum values that can be specified in the query according to the distribution of the data.

The next step is to perform a nearest neighbors search among all the dataset. Target is defined by the point that has been created, and the distances are calculated only considering the features given in the query. That is to say, if user has defined envelope and pitch, the distance in the search is computed only with the envelope and pitch. We are using Euclidean distance. The number of results returned by Gaia is set by a variable that can also be changed from the interface, but it is a constant number. Gaia always returns N most similar results according to the target, but there are two exceptions. First one has to do with the pitch. When one of the defined perceptual aspects is the pitch, there are at least two defined features: pitch and dissonance (see section 3.2.1). Pitch value is used in the nearest neighbor search, but dissonance is imposed later as a constrain. Gaia gives the ability to perform queries with filters, that is to say, to set a target and add other constrains using filters. This is how we implement the dissonance characteristic. Search returns the N closest values to the given pitch and that have a similar dissonance in a small range (± 0.2) around the given one. There might be the case that Gaia finds less than N results that satisfy dissonance constrain (therefore, search returns less than N results), however, this almost never happens. A similar thing is done with the tonal content perceptual aspect (the second exception). When users introduce specific tonal content target for the query, we understand that they are looking for sounds with a strong (identifiable) key. That is why we add the constrain that only results with key strength above 70% are returned.

Once the list of the results is generated, we compute the PCA considering all the features of the four perceptual aspects described in section 3.2.1. In this way we obtain the x and y coordinates that are used in the interface for the projection of the points (in "automatic" mode for sorting the results).

To finish with the query process, Python module returns a list with all filenames resulting of the nearest neighbors search, x and y coordinates from the PCA for each file and also the value of the four descriptors that can be used in the "direct mapping" mode for sorting the results (see section 3.3.1). In this way, the interface has immediate access to all this information at the same time of receiving the list of results. Other features are only accessible using the previously commented service "Get features".

3.5.3 Flex module

Regarding the implementation of the Flex module we will only present the general structure and show the main interactions that take place. We will not get into details because we think it is out of the scope of this report to explain how did we implemented particular aspects. Figure 3.12 shows a general block diagram of the main elements of the interface. What follows is a short description of these elements.

- **Interface:** this element contains the layout of the interface, the buttons, sliders and combo boxes that are editable by users. The interface also contains an instance of a "Results container" and a "Query panel". Communication with the server is also set up at this point along with the initialization of all variables that configure the interface.
- **Results container:** this is where the basic information of the results returned by the server is stored. This information consists in the name of the files, their PCA coordinates and the four descriptors used in the direct mapping mode for sorting the results. This is all the information needed to visualize the map. Every time a new query is performed, results container is cleared and filled up again with the new set of results. Results container is also in charge of updating the visualization of the SERP when users perform any action that requires it (zooming, dragging, selecting a result...). Results container contains N instances of "Sound point", which store the detailed information of each result.
- **Sound point:** this data structure is used to store all the specific information regarding one specific result. When a result is selected, sound point com-

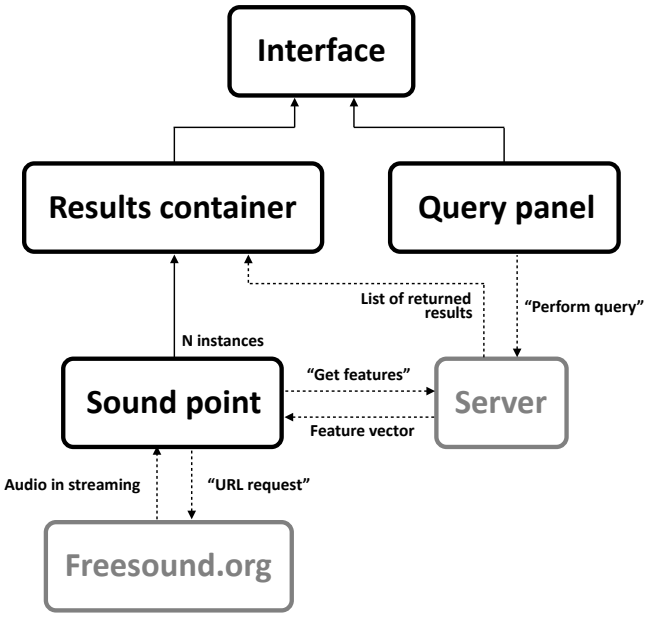

Figure 3.12 – General block diagram of the Flex application.

municates with the server and asks for its feature vector (the values of the descriptors corresponding to the four perceptual aspects that are defined). If a result is never selected, this information is never loaded in the interface because it is not needed. Sound point is also in charge of communicating with Freesound.org to reproduce the corresponding audio in streaming.

- **Query panel:** finally, the query panel is the element responsible of the query definition process. It allows to introduce the query target as explained in section 3.2, and later it communicates with the server to perform the query. The server returns the list of results directly to the results container. Query panel is also in charge of displaying the graphical representation of the perceptual aspects of a particular result when it is selected. In that case, it retrieves the information from the corresponding instance of sound point.

Chapter 4

Evaluation

4.1 Evaluation design

As a final step for the master thesis we have performed an evaluation of the prototype. We asked some people to perform certain tasks with the interface and analyzed their response to extract some conclusions.

4.1.1 Goals of the evaluation

Evaluation was designed to assess the performance of the evaluation regarding different aspects. Although many details could be analyzed, we focused on the following goals:

- **Search Engine Results Page:** we want to analyze how users understand the information given by the SERP and how they orientate given a set of sound results.
- **Query specification:** as another important aspect, we analyze the usage that users make of the query specification, which perceptual aspects are more useful related to the relevance of the results. As the interface is designed to find relevant sounds in a musical context, we evaluate the capability of users in finding such sounds giving a particular target.
- **Comparison with text-based search:** finally, we also perform a comparison between the text-based search interface *Freesound.org* (where sounds are described using tags, mostly regarding their *source*) and our prototype (where sounds are described by graphically defining their perceptual qualities).

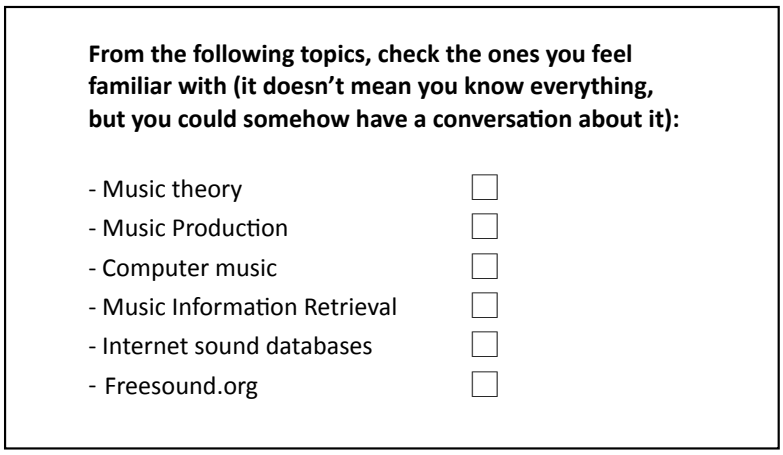

Figure 4.1 – Pre-test questionnaire.

4.1.2 Implementation

To analyze the previously commented goals we designed a test consisting in several parts that had to be completed by users, lasting around a total of 45-50 minutes per user. The test was carried on with 9 participants with different backgrounds (although all related to music). The idea was to gather some students of music technology and some musicians not necessarily familiarized with computer music. The different parts of the test were carried on as follows:

- **Pre-test questionnaire**: through this very simple questionnaire we pretended to collect information about users such as their experience in searching sounds in the web or familiarity with MIR concepts or music theory. Figure 4.1 shows that questionnaire.
- **Training stage**: as a second step, we gave participants a written description of the interface explaining how does it work, letting 10 minutes to freely explore it and encouraging them to make some questions if they had any doubt. In practice, almost no one used the written description and preferred an oral explanation.
- **Experimental tasks**: during this part of the evaluation, participants had to perform some tasks using the interface, and we collected some quantitative data by saving all their activity into a log file. The tasks they had to perform were divided in the three parts that are explained below:
  - **Part I**: in this part users were given a set of 60 results and were not able to perform new queries. The only thing they could do was to navigate among the results and listen to them. Among this results, a target

was randomly chosen and users could listen to it as many times as they wanted. The goal was to find that target sound among the set of results (locate the corresponding point in the map). When they found the target we considered that task was accomplished.

This process was repeated 6 times using three different type of maps (therefore, each map was used in 2 out of those 6 tasks). First type of map was build using the automatic sorting mode (PCA). For the second type we used the direct mapping mode, and users could choose which descriptors to assign in horizontal and vertical axis (as explained in section 3.3.1). Finally, third map used coordinates from the PCA but with the sounds randomized (we call it random mode). Result points were represented with the color and size mapped to their brightness and duration for all kind of maps.

We designed these tasks to analyze the usefulness of different map organizations and see how users orientate given a set of results. We monitored the number of sounds that users had to listen before finding the given target. We expected that by using direct and automatic maps users had to listen less results before finding the target.

- **Part II**: in the second part users had to perform 4 tasks. An example sound was given and they had to find as many similar sounds as they wanted, marking them as interesting. Each task had a different sound example, but were the same for all users. This sounds consisted in musical elements typically found in music productions: a drum kick, a single note (performed by a distorted organ), a single chord (a pad of a synthesizer) and a snare.

  We designed these tasks to analyze the kind of queries performed by the users in relation to the relevance of the results (similarity) given the example. Moreover, we also monitored the usage of the different sorting modes that users could freely change.

- **Part III**: finally, there was a third part consisting in 4 more tasks. In the same manner as in the previous part, users were given a target example and they had to find as many similar sounds as they wanted. The difference in that case was that 2 of this 4 tasks were performed with the text-based interface of *Freesound.org* instead of our prototype. Different targets were designed to represent natural sounds (easily describable by their source such as bells or cymbals) and artificial sounds (such as *glitches* or *filter sweeps*). Each user was presented a natural sound and had to

use the two interfaces (separately) to find the most similar sounds. Then, the same process was done with a artificial sound. The order in which the sounds were presented and the interfaces were used was randomized for each user.

We designed these tasks to perform a comparison with the text-based interface and see how useful are the interfaces for finding natural and artificial sounds. Regarding natural sounds, we expected better results (more similar) using the text-based interface, while regarding artificial sounds, we thought it would be better the graphical interface (because we think they are easier to describe with their perceptual characteristics).

- **Post-test questionnaire**: After the experimental tasks users were asked to fill another questionnaire with some general and some specific questions regarding their experience with the interface. Details on these questions can be seen in section 4.2.4.

4.2 Results analysis

In the following sections we show the results obtained with the experimental tasks (parts I, II and III) and present some conclusions. Finally we also comment on general qualitative results, based on the post-test questionnaire and other observations during the tests.

4.2.1 Part 1

Table 4.1 shows the percentage of tasks that were successfully accomplished by users according to the type of map. We consider that a task was not completed when users, after a certain time, decided to move to the next task because they could not find the target sound. As we can see, only using direct mode users were able to find the targets in all tasks. Random and automatic organization present less accuracy, with

	Automatic (PCA)	Direct mapping	Random map
Percentage found	77.78 %	100 %	88.89 %
Checked items	64.71	27.41	33.44

Table 4.1 – Percentage of accomplished tasks and average of checked items per map type.

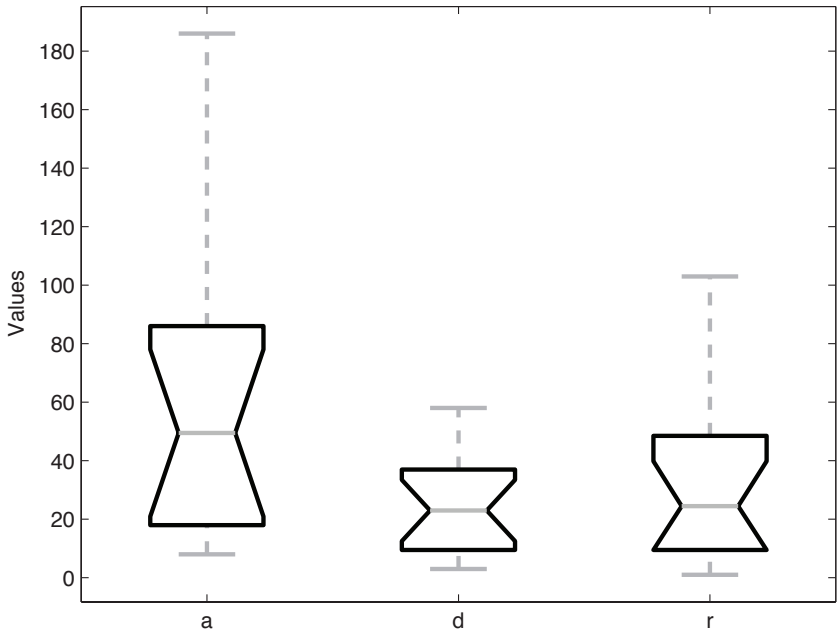

Figure 4.2 – Box plot of the number of consulted results according to map type (a = automatic, d = direct, r = random).

the second one obtaining the worst percentage. In the same table we also present the average of checked items (listened sounds) before finding the target. Again, results show that direct mapping was the organization that needed less consulted items before completing the task. It is closely followed by the random map (6 items more). Using automatic mode users needed to listen twice the number of results.

We did not analyze the time needed to accomplish the tasks because it was highly dependent on users attitude (some of them were quickly listening one sound after another while other were more calmed and the interaction was significantly slower) and on the length of listened results.

Figure 4.2 shows a box plot representing the number of checked items before finding the target in each kind of map. We performed an Anova test (within subjects variation) and obtained no statistical significance that results belong to different populations ($F = 2.18, \rho < 0.0894$). Furthermore, a multiple comparison of means shows that random mode results overlap both direct and automatic modes, but that those two are statistically separated between them (figure 4.3).

We did not find different results by analyzing subsets of users according to their background and experience (they all performed similar).

Considering these results we cannot support any strong conclusion, but we can observe a tendency which shows that direct mapping is the most useful map orga-

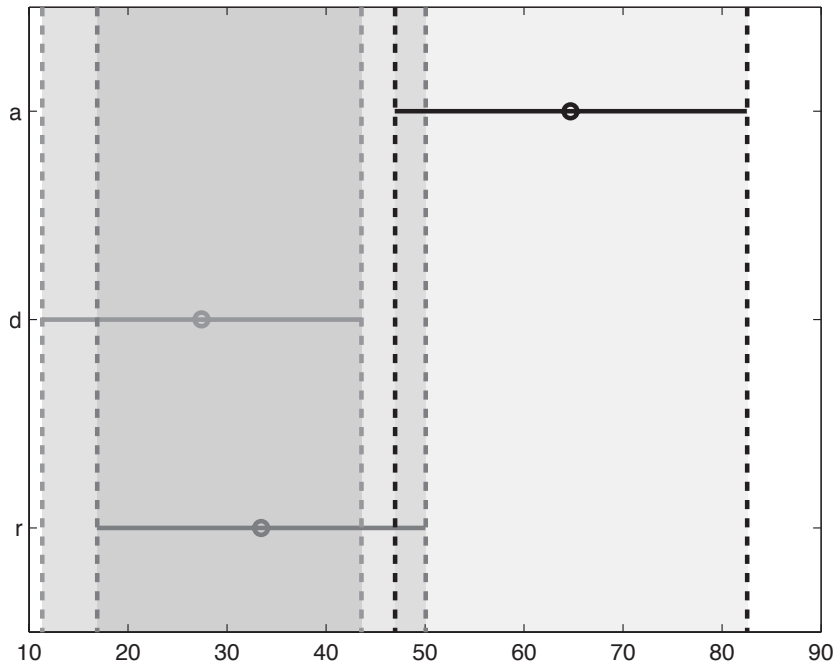

Figure 4.3 – Multiple comparison of the means for the number of consulted results according to map type (a = automatic, d = direct, r = random).

nization.

What surprises is that random mode results are closer to direct results than to automatic results. This can be due to the small amount of participants. There were some cases in which sounds were found in the very first clicks and this might distort the results (by coincidence). Nevertheless, in automatic organization users checked an average of 64 item per map, which is more than the total number of displayed results. This indicates that some of the results were listened twice or more. Considering that we already knew that PCA projection was not working properly, we think that those bad results were because users were trying to rely on that similarity organization and got confused (that could also be the reason why users insisted in checking again the results in certain regions of the map, because they heard some similar sound and were convinced that the target should be around).

Regarding percentage of tasks accomplished per map type, we can say that this strongly depended on the users. Some users decided to go the next task after a certain time had passed and they did not find the target, others just insisted until they found it. However, direct mode has an accuracy of 100% with the smaller average of checked sounds, which supports the tendency previously commented.

Finally, we can conclude that we did not get significant results but we observed some tendency. We expect that with an improvement of the automatic sorting and a new evaluation with more participants, clearer results could be obtained.

4.2.2 Part 2

In this part of the analysis we had to evaluate the relevance of the results marked as interesting in each task. Out initial approach was to compute the numerical distance between the feature vectors corresponding to each result and the target. However, we found out that this distance was not always reliable. It turned out that some sounds were similar by means of the numerical distance and others were completely different although represented a similar musical concept (that is to say, numerical distance was high but usability in a musical context was similar, *i.e.* same notes performed by different instruments). For this reason we manually labeled the relevance of each result with a scale from 1 (most similar) to 7 (most dissimilar), taking in account both perceptual and *conceptual* similarity. To perform this process, we randomized all the results for each task and manually labeled them without knowing to which user they belonged.

Using this distance, we see that best results have been obtained in the second task (musical note). First and third tasks have similar results (kick and chord) while snare has the biggest distance (table 4.2). Figure 4.4 shows a classification of the resulting distance for each task in discrete categories (from 1 to 2, 2 to 3,... and 6 to 7). In each category we can see the perceptual aspects that were used in the query definition (as a percentage taking in account all query definitions belonging to the tasks classified in the same category). Observing this figure we can identify some tendencies. Tonal descriptor is mostly used in the first category of distance (that is to say, when better results are obtained). Timbre descriptor increases in usage as results become worst. Finally, pitch and envelope are the main used descriptors specially in the distance range from 2 to 5 (although envelope is more or less equally used in all categories).

Task 1 (kick)	Task 2 (note)	Task 3 (chord)	Task 4 (snare)
3.3438	2.9706	3.5217	4.4348

Table 4.2 – Mean distance of the selected sounds per task (1 = similar, 7 = dissimilar).

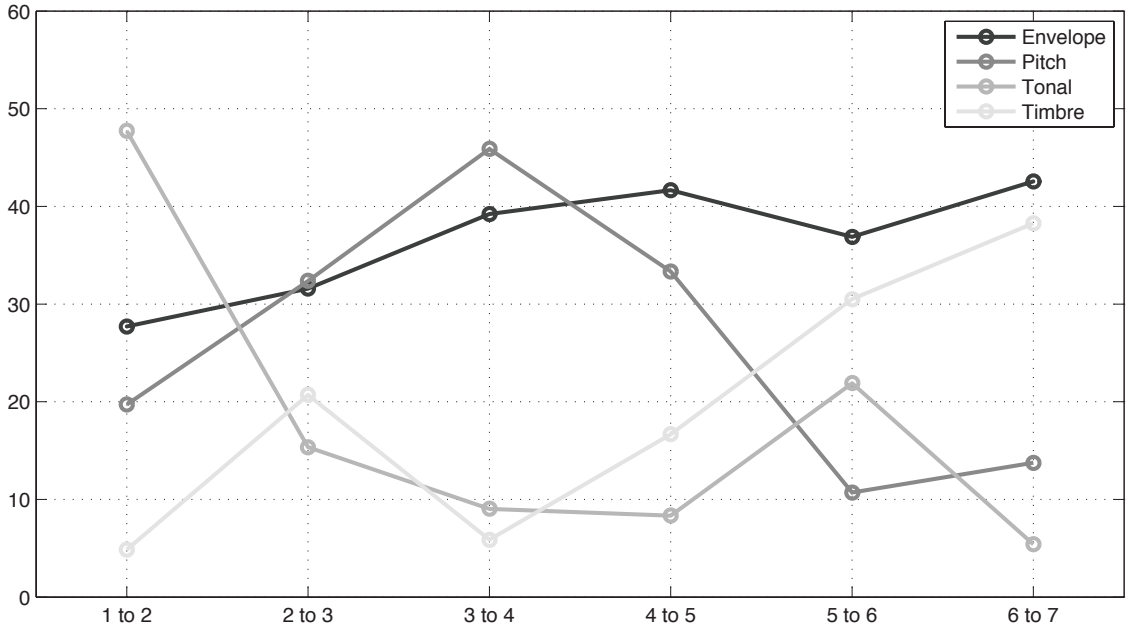

Figure 4.4 – Usage percentage of the perceptual aspects in the query definition according to distance (1 = similar, 7 = dissimilar).

Regarding the general usage of the sorting modes for the results map, figure 4.5 shows the percentages obtained analyzing log files in comparison with the scoring they obtained in the post-test questionnaire. In this figure we only show the descriptors used in the direct mapping mode, however, 62% of the time results were organized with the automatic map. This might probably be because it was the default option and some users did not mind to change it.

We do not observe relevant variability in the scorings of each mode (2% standard deviation), however, we observe a contour that is also preserved and emphasized in the usage according log files. Looking at this data, duration is the most useful descriptor to organize the results, and key is the less useful one. Pitch and brightness remain in a similar position (with the former being a bit higher).

Regarding the usage of perceptual aspects in the query definition, figure 4.6 shows the percentages obtained analyzing log files in comparison with the scoring they obtained in the post-test questionnaire. Here, the variability of the scoring is even less than in the previous figure (1.4% standard deviation) and the contour is not preserved in the usage according log files, which show that envelope is the most used descriptor for the search followed by pitch and finally timbre and tonal content.

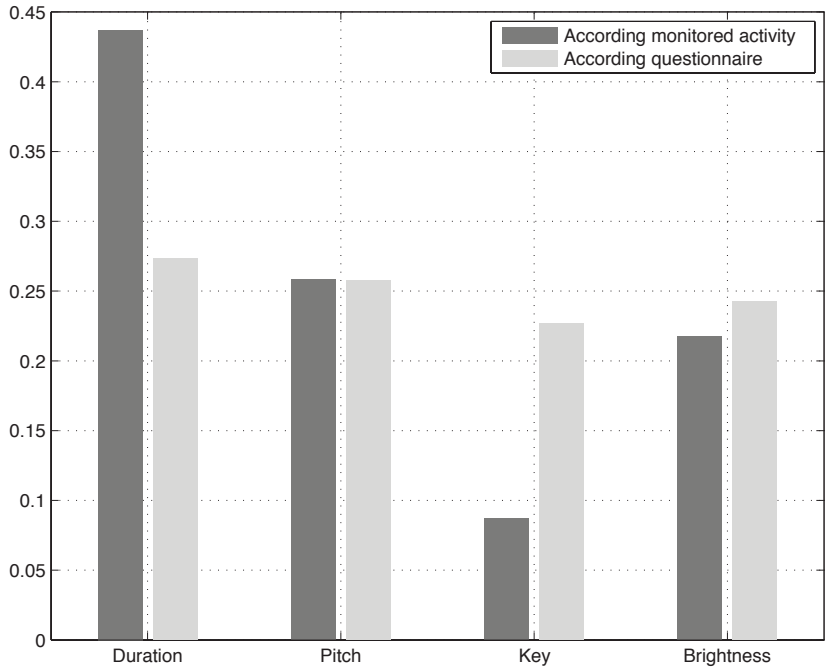

Figure 4.5 – Usage percentage of the descriptors in the direct mapping organization mode.

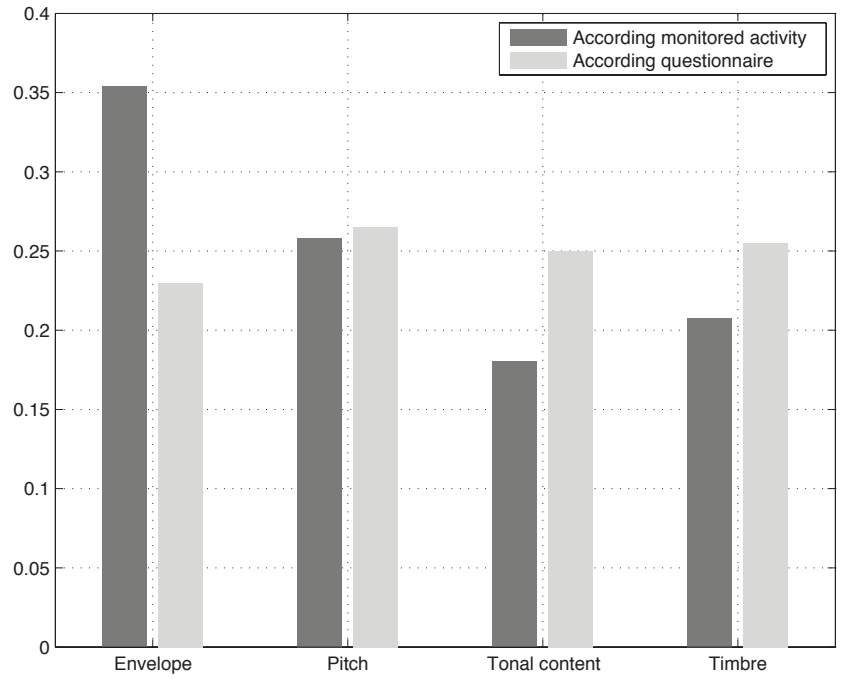

Figure 4.6 – Usage percentage of the perceptual aspects in the query definition.

Considering these results, we can suppose that the fourth task (snare sound) was the hardest to find probably because it could not by described by means of pitch or tonal content (those descriptors tend to gave good results). Moreover, the length of a snare sound can be similar to a lot of other sounds and, as a consequence, timbre was crucial in the query definition. It was difficult for users to understand which range of frequencies was present in the snare sound. Other sounds could be easily defined (note by the pitch, chord by the tonal content and kick by very low pitch or very low frequency content).

Best distance results were obtained when tonal information was more used in the query specification (almost 50%). This could be due to the specificity of this descriptor. If users were able to correctly specify the chord in task 3 (C, E, G), it was easy to come up with some sounds close to the target (C major chord, atmospheric pad). Aside from this case, tonal content was not used in other contexts (some users used it to reinforce the note when searching for a specific pitch, with good results tough).

Regarding the fact that distance range from 2 to 5 was mostly obtained using pitch and envelope, we think that this might be due to the simplicity of these descriptors, particularly in the case of pitch, but also for the envelope. Users tend to consider the envelope only for the duration of the sounds. They did not draw complex sound envelopes (may be at the first queries but not later) but tend to draw simple lines from the starting point (time 0) to the desired duration. Somehow they considered this descriptor as a one-dimensional perceptual aspect, like the pitch. Pitch and envelope worked quite well to give an overview of a sound (at least for the given tasks).

Timbre descriptor was thought to be really useful, but it turned out to be confusing for users. This is probably because it was difficult to identify the perceived frequency content with the corresponding Bark bands. There were some easy cases where it was easy to identify a very dark or very bright sound and then users correctly used Bark bands (only setting the first or the last bands), but all the range in the middle was confusing. It seems obvious that to be able to perform specific queries using this descriptor users need a certain experience with the concept of the descriptor but mainly with the interface. We expect that with a longer usage of the interface users could learn patterns of perceived timbres by observing the Bark bands of the obtained results. The fact that we did not observe differences in the performance of users familiarized with MIR and others that were not, supports the previous statement.

Finally, regarding the usage of the sorting modes, we think that the fact that automatic organization was the default option biased a bit users behavior. However,

if some of them did not mind to change the mode, this might be because they were not relying a lot in the position in the map but maybe only looking at the color and size of the points. Nevertheless, users that switched to the direct mapping mode, almost always used duration (45% of the time, taking in account that there were two axis, it means that one of them was almost always mapped to duration) in combination with either pitch or brightness. Key organization did not seem to be useful, probably because in the four targets, only the third (the chord) was expected to return results that could be successfully organized by key.

The fact that brightness was highly useful in the results organization but timbre was not in the query specification might be because brightness is a one-dimensional *simplification* of the timbre, and was easier to understand for users than the Bark bands. May be timbre could have been specified in the query just by setting the spectral centroid.

4.2.3 Part 3

For this part of the evaluation we also had to measure the relevance of the set of results chosen by each user in each task. For that purpose we followed the same process described in the previous section (a subjective scale being 1 the most similar and 7 the most dissimilar, performing an average with all selected results for each task).

Using this measurement, we computed the average distance obtained according to the type of sounds used as targets (natural or artificial) and the interface (text-based or graphical interface). Results are shown in table 4.3. We find that most relevant results are found when using text-based interface and looking for natural sounds. Following in relevance we find graphical interface and artificial sounds. Figure 4.7 shows the same information separated by participants.

As it was expected, looking for natural sounds was easily done with the text-based interface. Artificial sounds were also clearly easier to find with the graphical

	Freesound.org	Graphical interface
Natural	1.5	4.76
Artificial	5.32	3.41

Table 4.3 – Mean distance in the search of natural and artificial sounds with the text-based Freesound.org and the graphical interface (1 = similar, 7 = dissimilar).

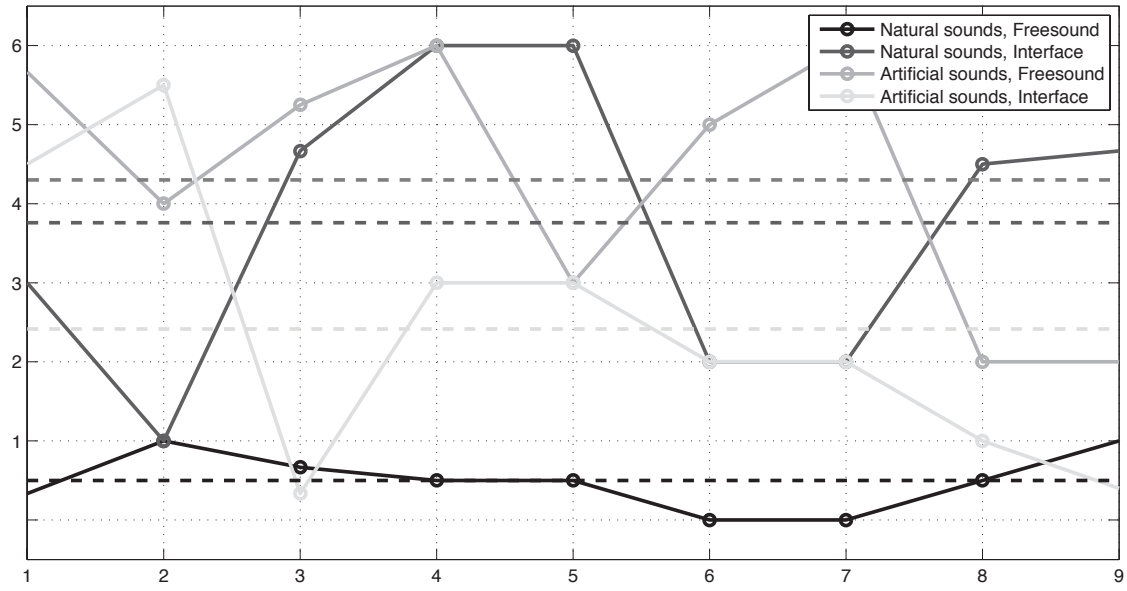

Figure 4.7 – Mean distance according to the type of sounds and the used interface.

interface. In the other cases, results wrere quite similar but higher than the previous ones. Those results are in accordance with informal comments that users made during the test. They pointed out (without asking) that it was easier to find sounds with text-based search when they knew how to textually describe it, while in the other cases more satisfying results were obtained with the graphical interface.

Aside from that, we observed that users selected and listened to more sounds per task when using the graphical interface. This was also expected because the graphical interface gives the facility to quickly navigate and listen the given results displayed in the map, while with the text-base search results are displayed as lists of sound files and that makes slower its accessibility.

4.2.4 Additional qualitative results

We already introduced some qualitative observations in the analysis of the experimental tasks. However in this section we will point out some general observations made while users were performing the test and some information gathered in the post-test questionnaire.

First of all, users had the tendency to perform very complex queries, but usually the ones that produced better results were the simplest ones.

The concept of dialogue between query and results was only explored a few times by users (use characteristics of the results to perform new queries). This might be

normal because they were not particularly encouraged to do that (for the tasks they had to accomplish) and probably it was a bit *advanced* usage of the interface that needed more practice. We believe that the use of this dialogue might help in learning the meaning of the descriptors (learning how to perform better queries), however with the evaluation we have performed we can not extract any conclusion on that.

Regarding the post-test questionnaire, the interface obtained good ratings. Users think they understood how it works and that it is not complex (although not anyone could use it, some specific knowledge is needed). Nevertheless, agreed in that provided functionalities could have been better integrated.

Punctuations regarding the query definition and the results representation are all positive, with an average of 5.2 over 7. Envelope definition has the lower rating among the perceptual aspects, and this is a bit controversial because it is the most used descriptor for the query definition. We think that the poor rating is due to the way in which envelope is introduced (not so practical) rather than in its utility for the query. Finally, bad results obtained with the automatic sorting of the results are reflected with a qualitative punctuation of 3.57 (below the center value, which would be 4).

Users also think that graphical interface can be better to explore the database in comparison to text-based approach, but that this depends mainly on the kind of sounds we are looking for (natural sounds, artificial sounds...).

Finally, all users coincided in that they enjoyed a lot using the interface and that it is an interesting tool. Some of them pointed out that it could be useful to add some "musical" functionalities such as being able to create loops or directly recording and adding new elements to the database, making it interesting as a performance instrument.

Chapter 5

Conclusions and future work

5.1 Conclusions

In this section we will comment various conclusions that have arisen during the whole development and evaluation process. We organize them in three categories regarding different aspects of the process.

Regarding System design

Perceptual aspects of the query definition are useful but somehow unbalanced. They do not have the same relevance and comprise different levels of detail. Moreover, the way in that information for the query is introduced is a bit complicated for the users. There is a lack of feedback that can make hard to understand the meaning of the descriptors. The case of the tonal content is particularly unpractical for setting a key (for example, a combo box list would be much easy to use).

Concerning results representation, we have seen that direct mapping of the descriptors is the most useful organization, probably because it is the easier to understand. Moreover, color and size mapping for the points in the map is simple, useful and understood by users. Dimensionality reduction do not provide satisfying results, but we think that this is because we should have needed more time to get it working properly. In fact, we did some tests in Matlab using MDS and Self-Organizing Maps, but it was difficult to implement these algorithms in our prototype and thus to be able to evaluate the benefits and drawbacks of different methods in the context of the interface. Although PCA organization seemed to confuse users (and was poorly rated), they insisted in trying to understand this *similarity* organization. We still think that with more research "similarity organization" should be a good solution.

The number of returned results in each query was set as a constant value, regardless of the distance of the results in respect to the target. Up to some point this is good because some unexpected yet interesting results are presented to users. However, sometimes these "extra" results are only noise and do not help at all.

Finally, the design of the layout and interaction of the interface represented a small amount of time in comparison to other aspects. There were too many aspects to consider and we left as less important in the development of the prototype.

Regarding System evaluation

We tried to evaluate relevant points regarding the orientation of our interface. However, there are so many other aspects that could have been evaluated. We did not obtain strong results, but we observed some tendencies and we are aware that we only had a small number of participants. Some of the tasks were difficult to analyze (specially in the second part) because there were so many variables affecting. For example, in the first part of the evaluation we concentrated in analyzing the map organization by not allowing users to perform any queries. However, in the second part we did not isolate the query panel and analyze the problems users could have with that interaction. As a consequence, when we find bad query specifications it is hard to distinguish if problems rely on the interaction or in the understanding of the descriptors.

At the end, from a qualitative point of view, all users reacted with enthusiasm and said the prototype is interesting and has a potential, which encourages us to continue working on it.

General conclusions

Due to the small amount of time we had both for the design and the implementation (this took much more time than we thought), a lot of ideas could not have been explored or we had to opt for simple solutions. If we had to start again, orientation would have been quite different (in the future work section we give ideas on that).

The biggest change in the concept of the interface could be to include the possibility to merge text-based and graphic query specification. We think that the interface would be much more useful if users could use words to define the query and graphically specify further constrains.

From another point of view, we think that an interface such this one (accessing a big sound database like Freesound), has a big creative potential that should be

explored. Just to give quick examples, simple and almost immediate applications would be to establish loops between returned results (similarly to Freesound Radio) or to map single results to the keys of a sampler (oriented to live performance).

Finally, we understand that this prototype is a proof-of-concept which shows that graphical query definition based on perceptual descriptions of sound can be successfully applied to audio databases and can be the object of further research.

5.2 Future work

To finish this report we propose some clear and immediate possible directions that could be researched to continue working and improve the interface.

- Improve the query definition by using more appropriate sound descriptions and better feedback. Investigate useful descriptions for sounds, that is to say, what do users really need to specify of a sound and what they don't. We would probably find different answers according to the kind of sounds users were looking for, and would probably have to perform categorizations.
- We could also investigate in the common semantic descriptions assigned to perceptual aspects of sound (particularly to timbre), and devise general patterns to be used in the query specification (*i.e.* having presets for *bright*, *mellow*, *dull* or *sharp* sounds).
- To improve feedback in the query specification we could try to introduce some audio cues. For example, in the pitch definition we could reproduce a tone with the current pitch while users were setting it. The same could be applied with the tonal content (reproducing a scale with the selected notes or a chord) or with the timbre (using filtered white noise for example). This could help users in understanding the query definition.
- Add the possibility to perform text-based queries. Considering this addition, interface should be redesigned from the beginning to exploit the combination of both search methods.
- Constrain the returned results by filtering out all the items that are not inside a numeric range around the target. This could help in better isolating good results, although some queries might not return any items.

- Try other methods for the dimensionality reduction that provide better organization. Not only the technique but also the feature vector used for this dimensionality reduction should be investigated. We observed that users tend to understand better simple mappings, so we could perform the dimensionality reduction using lower dimensional features, and this could also help in preserving the structure of the data in the two-dimensional projection.

- Achieve the concept of dynamic querying. That would require the results to be constantly updated as users were defining the query. In the prototype this was impossible due to implementation constrains, but we think in a future could be introduced and would be really useful to better understand the relation between the query and the obtained results.

- Perform a better evaluation with more subjects and a longer period of time to analyze the learning rate of the interface.

Bibliography

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Addison-Wesley Reading, MA, 1999.

[2] M.J. Bates. Where should the person stop and the information search interface start? *Information Processing & Management*, 26(5):575–591, 1990.

[3] M. Battermann, S. Heise, and J. Loviscach. Sonosketch: Querying sound effect databases through painting. *Proc. 126th Convention of the Audio Engineering Society*, 2009.

[4] T. Bergstrom and K. Karahalios. Seeing more: Visualizing audio cues. *Urbana*, 51:61801, 2007.

[5] E. Brazil and M. Fernström. Audio information browsing with the sonic browser. In *International Conference on Multiple Views in Exploratory Visualisation (CMV2003), London, UK*, 2003.

[6] E. Brazil, M. Fernström, G. Tzanetakis, and P. Cook. Enhancing sonic browsing using audio information retrieval. In *International Conference on Auditory Display ICAD-02, Kyoto, Japan*, 2002.

[7] P. Brossier, Q. Mary, E.E.P.E.R. Miranda, and M. Casey. Automatic annotation of musical audio for interactive applications. *Centre for Digital Music, Queen Mary University of London*, 2007.

[8] P. Cano, M. Koppenberger, P. Herrera, O. Celma, and V. Tarasov. Sound effects taxonomy management in production environments. In *Proc. AES 25th Int. Conf., London, UK*. Citeseer, 2004.

[9] Pedro Cano, Markus Koppenberger, Sylvain Le Groux, Perfecto Herrera, Julien Ricard, and Nicolas Wack. Knowledge and content-based audio retrieval using wordnet. *Institut de l'Audiovisual, Universitat Pompeu Fabra*, 2004.

[10] S.K. Card, T.P. Moran, and A. Newell. *The psychology of human-computer interaction*. CRC, 1983.

[11] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: current directions and future challenges. *Proceedings of the IEEE*, 96(4):668, 2008.

[12] C. Chen. Information visualization. *Information Visualization*, 1(1):1–4, 2002.

[13] M. Cooper, J. Foote, E. Pampalk, and G. Tzanetakis. Visualization in audio-based music information retrieval. *Computer Music Journal*, 30(2):42–62, 2006.

[14] A de Cheveigné and H Kawahara. Yin, a fundamental frequency estimator for speech and music. *Acoustical Society of America*, 111(4):1917–1930, 2002.

[15] W.W. Gaver. What in the world do we hear?: An ecological approach to auditory event perception. *Ecological Psychology*, 5(1):1–29, 1993.

[16] E. Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294, 2006.

[17] M. Goto. A chorus-section detecting method for musical audio signals. In *Proc. ICASSP*, volume 5, pages 437–440. Citeseer, 2003.

[18] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, 2005.

[19] J.M. Grey. Multidimensional perceptual scaling of musical timbres. *Journal of the Acoustical Society of America*, 61(5):1270–1277, 1977.

[20] S. Heise, M. Hlatky, and J. Loviscach. Soundtorch: Quick browsing in large audio collections. In *Proc. 125th Convention of the Audio Engineering Society*, 2008.

[21] R. Jeffries, J.R. Miller, C. Wharton, and K. Uyeda. User interface evaluation in the real world: a comparison of four techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 119–124. ACM, 1991.

[22] S. Jones. Graphical query specification and dynamic result previews for a digital library. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, page 151. ACM, 1998.

[23] D.A. Keim. Information visualization and visual data mining. *IEEE transactions on Visualization and Computer Graphics*, pages 1–8, 2002.

[24] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.

[25] C. Laurier and P. Herrera. Automatic detection of emotion in music: Interaction with emotionally sensitive machines. *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, pages 9–32, 2009.

[26] A. Mason, M. Evans, and A. Sheikh. Music information retrieval in broadcasting: Some visual applications. *BBC Research White Paper WHP*, 166, 2008.

[27] P. Morville and J. Callender. *Search Patterns*. Oreilly & Associates Inc, 2010.

[28] J.R. Olson and G.M. Olson. The growth of cognitive modeling in human-computer interaction since GOMS. *Human-Computer Interaction*, 5(2):221–265, 1990.

[29] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the tenth ACM international conference on Multimedia*, page 579. ACM, 2002.

[30] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2003.

[31] Saeffer. Pierre. Traité des objets musicaux. *Editions du Seuil. Paris. France*, 1966.

[32] G.E. Poliner, D.P.W. Ellis, A.F. Ehmann, E. Gómez, S. Streich, and B. Ong. Melody Transcription From Music-Audio: Approaches and Evaluation. *IEEE Transactions on Audio Speech and Language Processing*, 15(4):1247, 2007.

[33] J. Ricard and P. Herrera. Morphological sound description: Computational model and usability evaluation. In *Proc. 116th AES Convention, Berlin, Germany*. Citeseer, 2004.

[34] G. Roma, P. Herrera, and X. Serra. Freesound radio: supporting music creation by exploration of a sound database. 2009.

[35] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *SRM*, 154:1, 2006.

[36] D. Schwarz. The caterpillar system for data-driven concatenative sound synthesis. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, pages 135–140. Citeseer, 2003.

[37] D. Schwarz. *Data-driven concatenative sound synthesis*. PhD thesis, Université Paris, 2004.

[38] B. Shneiderman and C. Plaisant. *Designing the user interface*. Addison-Wesley Reading, MA, 1998.

[39] Ben Shneiderman. Dynamic queries for visual information seeking. *University of Maryland at College Park*, November 1994.

[40] W. Slawson. The Color of Sound: A Theoretical Study in Musical Timbre. *Music Theory Spectrum*, 3:132–141, 1981.

[41] E. Tanin, A. Lotem, I. Haddadin, B. Shneiderman, C. Plaisant, and L. Slaughter. Facilitating data exploration with query previews: a study of user performance and preference. *Behaviour & information technology*, 19(6):393–403, 2000.

[42] D. Temperley. What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception*, 17(1):65–100, 1999.

[43] G. Tzanetakis and P. Cook. 3d graphics tools for sound collections. In *Proc. COSTG6 Conference on Digital Audio Effects, DAFX*. Citeseer, 2000.

[44] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[45] D. Woods, E. Patterson, and E. Roth. Can we ever escape from data overload? A cognitive systems diagnosis. *Cognition, Technology & Work*, 4(1):22–36, 2002.

[46] W.S. Yeo and J. Berger. Application of raster scanning method to image sonification, sound visualization, sound analysis and synthesis. In *Proc of the 9th Int. Conference on Digital Audio Effects. Montreal, Canada*, September 2006.

[47] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Comput. Networks*, 31(11):1361–1374, 1999.

[48] J. Zhang, T. Shearer, G. Marchionini, M. Efron, and J. Elsas. Relation browser++: an information exploration and searching tool. In *Proceedings of the 2004 annual national conference on Digital government research*, page 2. Digital Government Society of North America, 2004.

[49] A. Zils and F. Pachet. Musical mosaicing. In *Proceedings of the International Conference on Digital Audio Effects*. Citeseer, 2001.