

# **MIRMaId: An interface for a content based Music Information Retrieval test-bed**

A DISSERTATION SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,  
FACULTY OF SCIENCE  
AT THE UNIVERSITY OF CAPE TOWN  
IN PARTIAL FULFILMENT  
OF THE REQUIREMENTS  
OF THE DEGREE  
OF

Masters in Philosophy in Information Technology

By  
Candice Lynn Cloete  
February 2006

Supervised by  
Dr. Hussein Suleman





# Acknowledgements

Like all dissertations, this dissertation could not be completed without the help and support of countless people over the past few years. I would therefore like to give my sincerest thanks for their help with this dissertation.

First of all I want to thank my supervisor Dr. Hussein Suleman, for his support, attention to detail and guidance. I cannot imagine having a better advisor.

Next I want to thank Mr. Donald Cook for his guidance, as well as graciously allowing me to use some of his contacts in the Computer Music field.

I also want to thank people involved in the Music Technology Group at Stellenbosch University and particularly Mr. Theo Herbst, for helping me procure test subjects and for answering many questions.

I also want to thank the past and present members of the Advanced Information Management laboratory at UCT, for their support, and other individuals who acted as patient test subjects.

I would also like to thank the past and present students from the Masters in Information Technology group, for their help, in particular Samuel King and Victor Katoma, for their support and friendship.

I would also like to thank the National Research Foundation for their research grant.

I would like to thank my parents Windsor Cloete and Rachel Terblanche, for their patience and support as well as Freddy Taggar, for his support and my brother Kevin Cloete, for keeping me laughing through many frustrating moments.

# Abstract

Music Information Retrieval (MIR) is the interdisciplinary science of retrieving information from music and includes influences from different areas, like music perception and cognition, music analysis, signal processing, music indexing and information retrieval [Futrelle & Downie, 2003] .

To produce the most efficient MIR systems, test-beds are commonly used to test different combinations of parameters against each other. The purpose of this dissertation was to investigate the composition of algorithms for MIR systems by constructing an interface that could form part of a test-bed. It differs from other interfaces and frameworks that are used in MIR test-beds because it is focused on small scale test-beds.

MIRMaid is an acronym for Music Information Retrieval Modular aid and is an interface that allows different content based retrieval tasks to be compared against each other to find optimal combinations of retrieval parameters for specialised problem domains.

The dissertation describes the process of how the MIRMaid interface was developed, modified and refined.

A big challenge was to design the user experiments in a way that considered potential users of the interface while using the test subjects I had at my disposal. I decided to use the simplest queries to highlight basic similarities between novice and potential expert users. The performance of the interface was judged by user ratings on a questionnaire. The interface performed reasonably well with expert users and novice users. Despite these results there were a few interesting observations that were returned from the user experiments related to the experiment design and the task explanations.

Some suggestions are also provided for extending the interface to allow it to be used with other types of data. The possibility is also investigated for using the interface as a tool for simplifying the process of integrating modules from different sources.

## *Keywords*

*Music Information Retrieval, test-bed, interoperability, interface, audio content extraction*

# Index

|  |    |
|--|----|
| Acknowledgements.....  | i  |
| Abstract.....  | ii |
| Chapter 1.....   | 1  |
| 1 Introduction.....  | 1  |
| 1.1 Problem Statement.....                                     | 1  |
| 1.2 The Solution.....  | 1  |
| 1.3 Thesis Structure.....                                      | 2  |
| Chapter 2.....   | 4  |
| 2 Theoretical Background.....                                  | 4  |
| 2.1 Overview.....  | 4  |
| 2.2 Data Mining and Musical Digital Libraries.....             | 4  |
| 2.2.1 Musical Digital Libraries.....                           | 4  |
| 2.2.2 Test-beds.....   | 5  |
| 2.3 The Structure of Digital Audio Data.....                   | 5  |
| 2.3.1 Sampling.....  | 5  |
| 2.3.2 Quantisation noise and resolution.....                   | 6  |
| 2.4 Psychoacoustics, sound perception and music cognition..... | 7  |
| 2.5 Basic Signal Processing operations.....                    | 8  |
| 2.5.1 Fundamental Frequency estimation.....                    | 8  |
| 2.5.2 Event Detection and Windowing.....                       | 9  |
| 2.5.3 Feature extraction.....                                  | 9  |
| 2.5.4 Matching.....  | 10 |
| 2.5.5 Transcription Models.....                                | 11 |
| 2.5.5.1 Set based models.....                                  | 11 |
| 2.5.5.2 Hidden Markov Models.....                              | 12 |
| 2.5.5.3 Audio Fingerprinting.....                              | 12 |
| 2.5.5.4 Self-Organising Maps.....                              | 13 |
| 2.6 Summary.....   | 13 |
| Chapter 3.....   | 14 |
| 3 Frameworks and Toolkits.....                                 | 14 |
| 3.1 Overview.....  | 14 |
| 3.2 Information Retrieval Frameworks.....                      | 14 |
| 3.2.1 CLAM.....  | 14 |
| 3.2.2 M2K.....   | 16 |
| 3.2.3 MARSYAS.....   | 18 |
| 3.2.4 MUSART.....  | 18 |
| 3.3 Music processing languages.....                            | 19 |
| 3.3.1 Matab.....   | 19 |
| 3.3.2 Octave.....  | 21 |
| 3.3.3 Labwindows.....  | 22 |
| 3.3.4 Nyquist.....   | 22 |
| 3.4 External Libraries.....                                    | 23 |
| 3.4.1 Machine learning libraries.....                          | 23 |
| 3.4.2 Music processing libraries.....                          | 23 |
| 3.4.3 Visualisation.....                                       | 24 |
| 3.5 Other Tools.....   | 24 |
| 3.5.1 Sphinx-3.....  | 24 |
| 3.5.2 WEKA.....  | 24 |
| 3.6 Summary.....   | 25 |
| Chapter 4.....   | 26 |
| 4 The MIRMaId Interface.....                                   | 26 |
| 4.1 Overview.....  | 26 |
| 4.2 Structure.....   | 26 |
| 4.2.1 Test-bed Structure.....                                  | 26 |

|           |   |    |
|-----------|---|----|
| 4.2.2     | The repositories.....   | 26 |
| 4.2.3     | Modules.....  | 27 |
| 4.3       | The Interface.....  | 28 |
| 4.3.1     | Design Goals.....   | 28 |
| 4.3.2     | Interface Elements.....   | 29 |
| 4.3.2.1   | The "Choose a repository" component.....  | 29 |
| 4.3.2.2   | The "Select Transformations" component.....                                       | 30 |
| 4.3.2.3   | Adding Transformations Frame.....   | 31 |
| 4.3.2.4   | Use case Control component.....   | 32 |
| 4.3.2.5   | Sound loading/recording component.....  | 33 |
| 4.3.2.6   | Matching/Evaluate/Execute the query.....  | 34 |
| 4.3.2.7   | Presentation of Results.....  | 34 |
| 4.3.3     | Interface development.....  | 36 |
| 4.3.4     | Strategic positioning of the framework.....                                       | 36 |
| 4.3.5     | Interviews.....   | 36 |
| 4.3.6     | Improvements on the interface.....  | 39 |
| 4.4       | Description of development tools used.....  | 40 |
| 4.5       | Summary.....  | 40 |
| Chapter 5 | .....   | 41 |
| 5         | Experiment and Questionnaire Design.....  | 41 |
| 5.1       | Overview.....   | 41 |
| 5.2       | Population Selection.....   | 41 |
| 5.3       | Experiment Design.....  | 42 |
| 5.4       | The Tasks.....  | 42 |
| 5.5       | Arranging outcomes from the interface.....  | 43 |
| 5.6       | Questionnaire Design.....   | 43 |
| 5.6.1     | Subject profile.....  | 44 |
| 5.6.2     | Interface and task based questions.....   | 44 |
| 5.7       | Justification for not using time measurement as an evaluation tool.....           | 45 |
| 5.8       | User observation.....   | 45 |
| 5.9       | Confounding variables.....  | 46 |
| 5.10      | Summary.....  | 46 |
| Chapter 6 | .....   | 47 |
| 6         | Data Analysis and Results of the Experiment.....                                  | 47 |
| 6.1       | Overview.....   | 47 |
| 6.2       | Sample population analysis.....   | 47 |
| 6.3       | Results Summary.....  | 49 |
| 6.4       | Discrepancies between results from task based and interface based statements..... | 50 |
| 6.5       | Evaluation of design results.....   | 51 |
| 6.5.1     | Measurement of design goals.....  | 51 |
| 6.5.1.1   | Adequacy.....   | 52 |
| 6.5.1.2   | Simplicity.....   | 53 |
| 6.5.1.2.1 | Intutiveness.....   | 53 |
| 6.5.1.2.2 | Learnability.....   | 54 |
| 6.5.1.3   | Usability.....  | 54 |
| 6.5.1.3.1 | Layout.....   | 55 |
| 6.5.1.3.2 | Navigation.....   | 56 |
| 6.6       | Problems with the MIRMaid interface.....  | 57 |
| 6.6.1     | Navigation.....   | 57 |
| 6.6.2     | General layout and operation logic.....   | 58 |
| 6.6.3     | Controlling query options.....  | 59 |
| 6.6.4     | Finding and Loading Sound Clips.....  | 59 |
| 6.7       | Summary.....  | 60 |
| Chapter 7 | .....   | 61 |
| 7         | Future Work.....  | 61 |
| 7.1       | Overview.....   | 61 |
| 7.2       | Interface Enhancements.....   | 61 |

|  |    |
|--|----|
| 7.4 Future testing of the test-bed and the interface.....    | 65 |
| 7.5 Summary.....   | 66 |
| Chapter 8.....   | 67 |
| 8. Conclusion.....   | 67 |
| Appendices.....  | 69 |
| A1 Task explanation used during the project.....             | 69 |
| A2 Instructions to the first task used in the interface..... | 70 |
| A3 Instructions to the second task used in the project ..... | 71 |
| A3 Questionnaire used in the project .....                   | 72 |
| Bibliography.....  | 74 |

## List of Tables

|  |    |
|--|----|
| Table 6.1 Summary of user profile in terms of music and computer training..... | 48 |
| Table 6.2 Results summary from the questionnaire test subjects.....            | 49 |

## List of Figures

|   |    |
|---|----|
| Figure 3.1 : Image of a CLAM flow control schedule.....   | 15 |
| Figure 3.2: Image of M2K workspace with flow control network.....                                     | 17 |
| Figure 3.3 : Image of the MATLAB workspace.....   | 20 |
| Figure 3.4 : Image of the Octave environment.....   | 21 |
| Figure 3.5 : Image of the labwindows environment.....   | 22 |
| <br>  |    |
| Figure 4.1 : Image of the "Choose a repository 11 element within the interface.....                   | 29 |
| Figure 4.2 : Image of the choosing transformations element within the interface.....                  | 30 |
| Figure 4.3 : Image of the adding transformation element.....  | 31 |
| Figure 4.4 : Image of the control form.....   | 32 |
| Figure 4.5 : Image of the sound recording element within the interface.....                           | 33 |
| Figure 4.6 : Image of the matching element within the interface.....                                  | 34 |
| Figure 4.7 : Image of the results screen.....   | 35 |
| Figure 4.8 : Image of the second results screen.....  | 35 |
| Figure 4.9 : Image of the main screen of the first interface.....                                     | 37 |
| Figure 4.10: Picture of the main screen of the interface after the interview.....                     | 39 |
| <br>  |    |
| Figure 6.1: Population Composition.....   | 47 |
| Figure 6.2: Indicates the percentage of subjects who could complete the task that they were set..     | 52 |
| Figure 6.3 : Graph of the responses from the layout question in the questionnaire.....                | 53 |
| Figure 6.4 : Graph of the responses from the navigation question in the questionnaire.....            | 54 |
| Figure 6.5 : Graph of the responses from the intuitiveness question in the questionnaire.....         | 55 |
| Figure 6.6 : Graph of responses from the question if the interface was difficult to learn or not..... | 56 |
| Figure 6.7: This figure illustrates the current navigation path of the current interface.....         | 57 |
| <br>  |    |
| Figure 7.1 : Image of how the interface could look like in the future.....                            | 61 |
| Figure 7.2 : New navigational path over one frame.....  | 62 |



# Chapter 1

## *1 Introduction*

### **1.1 Problem Statement**

Over the last ten years the discipline of Music Information Retrieval (MIR) has grown very fast, mostly without consensus on uniform data representations, common evaluation standards and common guidelines for interaction amongst different MIR frameworks. As a result there are many tools, music processing frameworks, test-beds, synthesis libraries and synthesis languages built to aid information retrieval varying in scope and language implementation that are unable to interact with each other directly and effectively.

Most of the frameworks designed for audio signal processing attempt to be comprehensive solutions but result in many core signal processing classes being duplicated across different frameworks. The differences in implementation are either due to internal structural differences in frameworks or that classes are implemented in different languages. At the same time there is only minimal support provided for including modules and classes from other frameworks.

Each framework has its own rules for accessing data, using storage and presenting file formats. There was no freedom to combine different objects from different locations without having to convert them first to another form manually or translating the module into the format that the framework accepts. This situation got to a point that compiling and running even the most simple programs in many frameworks became very complex. This situation also makes it difficult to test different MIR strategies used in different MIR systems.

### **1.2 The Solution**

A comprehensive study would tackle the problem of interoperability between different music processing frameworks within three areas: 1) modular interoperability, how modules from one framework can be directly imported into other frameworks without any internal changes in the module; 2) data interoperability, how to package and transport annotated audio data so it is archive, processing and framework neutral, without losing information or creating problems when

processing data and 3) framework integration, how one framework can access classes and tools in other frameworks without having to manually import classes from one framework to another by accessing one framework through the command line.

Within the context of these problems we decided to concentrate on module integration. This involved creating an interface for a content based Music Information Retrieval test-bed to investigate the compositions of algorithms for music manipulation in MIR systems simplifying the process of integrating modules successfully from different sources.

The future work chapter also explores the option of embedding the interface into a proposed International Music Information Retrieval Laboratory (IMIRSEL) [Downie, 2003] but not as a rival to bigger and well established evaluation frameworks, like M2K, as this interface is more geared towards handling small specialised repositories.

The dissertation expands on possibilities for extending the MIR frameworks and covers some scenarios of what could happen if the test-bed would include different types of data, other than audio data. It also proposes automatic testing of different combinations of test metrics for the same retrieval task and the same set of data against each other without having to do it manually.

This project also gives suggestions on how modules from different frameworks can be combined and different combinations of modules tested against each other.

## **1.3 Thesis Structure**

The first chapter and second chapter gives an overview of concepts, test-beds and frameworks that are currently used in music information retrieval, and attempts to justify the existence of the interface by relating it to other projects. It also discusses similarities, differences and problems in different projects.

The third chapter describes the interface development process and gives a description of how the interface works.

The fourth chapter deals with how user experiments were designed to test the interface that was built. First the justification behind the population selection is given and then the procedure for testing the interface was explained. This chapter also gives some explanations on how the questionnaire was constructed.

The fifth chapter presents the results that were returned from the experiments and gives details on the overall working of the interface.

The last chapter gives some suggestions on how the interface can be extended in different ways. It also presents suggestions for how a production quality test-bed can be implemented.

# Chapter 2

## ***2 Theoretical Background***

### **2.1 Overview**

This chapter gives an overview of signal processing, signal analysis, audio data and music digital libraries. It also lays a conceptual basis for discussing the test-bed and other related concepts in the rest of this thesis.

### **2.2 Data Mining and Musical Digital Libraries**

#### ***2.2.1 Musical Digital Libraries***

Musical Digital Libraries are a type of Digital Libraries that contain music in different formats. These include sheet music scores, bibliographic data, metadata, audio files and event based forms, like MIDI. There are very few collections that are both extensive and public, due to copyright restraints. One solution is to use music in the public domain or those published under the Creative Commons License agreement. The other solution, if copyrighted clips are necessary, is to return audio characteristics back from queries instead of sound clips. If the characteristics were re-combined it would make a reasonable but very low quality reproduction of the sound clip [Typke,2004]. Over the last few years copyrighted collections were used less often. One criticism against using music under Creative Commons Licences are that the songs are less known than their commercial counterparts and consequently less useful in query by humming tasks.

Most of the digital libraries and repositories available in the past retrieved digital audio records by querying the repositories by metadata, like composer, song title, performing artists or the publication date, indexes or text queries.

Currently there are music digital libraries that query a repository on the content audio file itself as is done in the Medlex/Greentstone project [Bainbridge et al. 2004] and other projects. These projects allow for queries to be added via text, event based data (like MIDI), sound clips and vocal queries (humming). Vocal queries are either done by matching it directly against audio input, irrespective of audio format or compression status.

Most tools available for querying datasets use a variation of converting the original audio signal first to either a symbolic form, a transformed form, or a preprocessed form.

After this matching algorithms can be applied and matching performed. Examples of this can be found in [Sandler,2001], [Batille & Cano, 2000] and others.

### **2.2.2 Test-beds**

A test-bed is an environment in which different theories can be rigorously tested and experimented with. Through this process successful tools can be identified and deployed in music digital libraries and in music content based search engines [UIUC DU Glossary,1998].

A test bed contains raw and modified data. Each song has associated sets of data, each version of the data having been processed by different transformations. Test beds also contain software tools, repositories and other tools to access, evaluate and manipulate the data in the test bed. On occasion test beds also have execution environments configured for testing.

There has been a lot of talk about creating a big unified test bed across different continents and research labs, with controlled access and strong security to convince companies to part with their copyrighted collections of music, like the ones that are available for other information retrieval disciplines, e.g. TREC for video data. The IMIRSEL (International Music Information Retrieval Systems Evaluation Laboratory) and MIREX projects will be discussed in the next chapter.

## **2.3 The Structure of Digital Audio Data**

To get Digital Audio Data from a mechanical sound wave an analog-to-digital converter is used. The analog-to-digital converter converts sound waves into a digital form which is then stored in a file. This process is commonly referred to as sampling [Steiglitz,1996].

### **2.3.1 Sampling**

Sampling is the act of converting time from continuous to discrete quantities by taking snapshots (called samples) of an incoming signal at set intervals and putting the result together to form a discrete signal. The sample rate (sampling frequency) is the rate at which samples are generated

over the course of one sinusoidal wave in one second and is measured in Hertz (Hz).

To accurately sample a continuous signal, the sampling rate must be at least twice as high as the value of the highest frequency present in the signal needing to be sampled. Not doing this will result in aliasing in which a frequency of a sample can be ambiguous.

The common sampling rate used in sound synthesis programs is 40 100Hz, because it caters for the upper limit of human hearing, but conserves sound fidelity and conserves computer memory.

### **2.3.2 Quantisation noise and resolution**

The higher the resolution of the sound the better the quality of the sound. The resolution depends on the size used for the word to represent the sample. If the resolution is too high you have what is called quantisation noise. This adds to the random noise that was already present in the original analogue signal. The audio data is stored as a sequence of bits approximating the signal.

The simplest way in which audio data is stored is as a sequence of bits, that is not altered in any way, after analogue to digital conversion. This is commonly referred to as raw audio data. Raw audio data is stored in a file that specifies other information, like the data format and the resolution/bit rate of the sampled sound.

The bit rate is the number of bits that are used to describe a single sample. This has an effect on the the fidelity of the sample, the dynamic range that can be achieved and how accurately the sound can be reproduced from its analogue form [Sun Microsystems, 2000].

Data formats tell you how to interpret raw sampled audio data. The samples can either be obtained by reading a file, or samples can be captured using a microphone input. Information that data formats can contain are the number of bits in the sample rate, the number of channels, the Frame rate, the Frame size(in bytes) and the byte order [Sun Microsystems, 2000].

In order for a sound to be captured or played back by any device, the data format of the sound you are capturing or playing needs to be specified [Sun Microsystems, 2000].

File formats specify the structure of a file and include information on the format of data in the file. File formats also include descriptive information. File formats differ from one another in their structure [Sun Microsystems, 2000].

Raw audio formats are typically based on open formats and can almost be universally played by all

audio applications irrespective of their operating system. These include the Microsoft's .wav format, Sun's .au format for UNIX's and Apple's AIFF format. These file formats store mostly uncompressed PCM-encoded raw audio signals in a single binary file [Reiss & Sandler 2004].

Compressed formats can either be done without loss of data, but there is both a computational and a size cost involved, or lossy compression where you lose some of the information when encoding the data. This is important for information retrieval, as encoding distorts the original saved form in the file. By preprocessing and low frequency sampling, the signal becomes drastically modified and loss of the stereo image can cause even robust similarity measures to fail [Reiss & Sandler 2004].

Exchange formats and wrappers encapsulate audio data. These wrappers carry custom metadata attributes and audio data in a variety of formats. These wrappers can also allow annotations to be associated with resources like the original sound clip via bindings/components.

XML can be used as a structural wrapper for music data, to give additional information about the sound wave besides the file format, like metadata or information about the program that created the wave. XML is also a convenient way to describe low level music descriptions with the help of the MPEG-7 audio standard description tools. These description tools help to describe music and other multimedia content.

There are two parts to the MPEG-7 audio standard - the first is the descriptors, defining the syntax and the semantics of each feature together with the description schemes that define the relationship between components, semantics and syntax. The second is the description definition language, which ties back to XML since XML is used to textually represent content descriptions [Zoja, Zhou, Mattavelli, 2001 ].

## **2.4 Psychoacoustics, sound perception and music cognition**

Music has five different facets that can be distinguished by the brain as making up a unique identity of a piece of music. The most important factors in listening to music are the timbre, pitch and duration facets. This is why these facets are also the most commonly used when extracting salient data from audio recordings.

The timbre facet refers to the feature that allows one to distinguish between two sounds that are equal in pitch, loudness, and subjective duration. The temporal facet concerns itself with the duration of musical events. The pitch facet is generally defined as the perceived quality of a sound,

which is the number of oscillations per second [Downie 2003]. The Harmonic facet allows the brain to distinguish between pitches when two or more pitches sound at the same time.

It is possible for both humans and computers to separate and distinguish between these different facets in monophonic queries. The problem comes in if the song displays polyphony. Polyphony occurs when multiple monophonic signals are present in one audio channel.

It is easy for the brain and the ear to make value judgements and distinguish between the separate facets within polyphonic melodies when many sound sources enter the ear. This is not the case with computers. The implications are that different signal processing operations are necessary in separating polyphonic signals to allow value judgements to be made in terms of the five facets and by extension retrieval of musical data from a database based on sound.

## **2.5 Basic Signal Processing operations**

Basic Signal Processing operations are important for extracting relevant characteristics from raw audio data.

### ***2.5.1 Fundamental Frequency estimation***

Fundamental Frequency (F0) estimation is an important extracting conceptually relevant characteristics from data, like loudness, rhythm and pitch.

Fundamental Frequency is the name given for the perceived pitch of a periodic sound [Steiglitz, 1996]. Fundamental Frequency estimation is the process of analysing an acoustic signal to estimate the predominant fundamental frequencies within a mix of signals.

For information retrieval this process is important, since it is used in pitch tracking and transcription for both monophonic and polyphonic audio signals. F<sub>0</sub> is also used for separating different "voices" where the entire signal is mixed in one channel of polyphonic signals.

There are four basic groups of F0 estimation methods; these are Time-domain frequency estimation, spectral pattern matching, frequency domain periodicity estimation and Auditory motivated methods [Klapuri, 2004].



Many Time-domain frequency estimation methods use Autocorrelation Function (ACF) based algorithms. In autocorrelation the maximum value in ACF is taken as the  $1/F_0$  period. It is used in polyphonic retrieval which is based on probabilistic time inference methods and other methods that use pitch as the main determinant for estimating matching methods [Klapuri, 2004].

Spectral interval based pattern matching is based on the periodic magnitude spectrum of harmonic sounds. It works better for sounds that exhibit inharmonicity, as intervals do not remain constant but are more stable [Klapuri, 2004].

Auditory motivated methods use human auditory perception as a template for how computer systems should perform  $F_0$  estimation and by extension pitch extraction. For each stage of human hearing there is a process that simulates the functionality of the ear. A criticism against auditory models is that it can be computationally expensive, because analysis needs to be carried out using multiple channel auditory filterbanks [Karjalainen & Tolonen, 1999].

An important issue in fundamental frequency detection is that sometimes one predominant frequency within a sound wave frame cannot be identified. This causes complications for systems that do sinusoidal separation automatically.

### ***2.5.2 Event Detection and Windowing***

Windowing is a way of segmenting audio data into notes by using event detection. The spikes in an amplitude envelope is used to detect if a musically relevant event occurred [Steiglitz, 1996], for example, if a note is played by an instrument or sung.

Data files are sometimes windowed into overlapping frames, with each frame representing one event so that there is only one distinguishable musical event per frame. There are different rules for segmenting musical data, and it depends on what type of processing is performed on the data and what the required size of the envelopes is.

### ***2.5.3 Feature extraction***

The biggest challenge in feature extraction is to get the most efficient and fault tolerant error models to take out the effect of human error and other anomalies when extracting features from

audio clips and human singing queries.

Originally most matching systems relied only on melodic contour information to compute feature vectors. One of the first pitch tracking algorithms implemented was pitch extraction by finding the peak of the autocorrelation function of the signal, using prominent peaks in the signal spectrum to apply autocorrelation algorithms to it [Haus, Pollastri, 2001]. This popular method was error prone due to the note segmentation processes. [Zhu & Shasha 2000]. The four other most common approaches to extracting pitch data for matching was to compare profiles of pitch direction, pitch contours, pitch-event strings or intervallic contours [Selfridge-Field, 1998].

The research was extended by converting sung queries into temporal data. Acoustic information is converted into relative intervals and used in making feature vectors. [Kosugi et.al. 2000] Another purely temporal solution is to use a time series database approach, which involved treating music as a time series. This allows for the use of well developed techniques from time series databases to index the music for fast similarity queries [Zhu & Shasha 2000]. Most recent features used, include various combinations of temporal data features and pitch duration pairs [Haus & Pollastri 2001].

#### **2.5.4 Matching**

Matching is the comparison of two feature sets against each other to see how similar they are.

After extracting perceptually relevant features from frame segments, distance functions are applied to relevant information. At the moment, this is the main way of matching temporal, harmonic and pitch data, or a combinations of these features [Typke, 2004]. There are many methods and concepts adapted to audio feature matching from the video and text retrieval fields, as well as from conventional music notation and symbolic data retrieval.

The two main methods for matching are exact matches and transposed matches. In an exact match, specific pitch information is matched, and is the main matching method used by audio fingerprinting and other brute force transcription models. In transposed matching, intervallic information is used to match records against each other and returns more results but less accurate matches than with exact matching.

There are different types of transposed matching methods associated with different variations and anomalies that happen in queries. These include: Matching with deletions, repetition identification,

overlapping repetition identification, transformed matching, distributed matching, chord recognition, approximate matching and evolution detection, where the search pattern tracks gradual change of the feature or feature set that is being matched against [Crawford & Iliopoulos, 1998].

### **2.5.5 Transcription Models**

Transcription is the act of transforming an acoustic signal into a form from which musical parameters can be extracted. The transcription methodology and methods applied to extracted data is directly dependent on the original form of the signal and the method that will be used for matching the data.

There are a wide range of transcription models, explaining how data should be extracted, transcribed and then matched. These include N-grammed models in which music is transcribed to N-grammed words according to different formulae. N-grammed models allow text retrieval methods to be used on musical data. Hidden Markov Models, in which different sets of data is extracted from the same audio data and then combined to match different records statistically, auditory models and various brute force methods of matching queries to records in a repository.

#### **2.5.5.1 Set based models**

Set based methods are used both for feature extraction of notated and audio data. These methods use feature extractors to convert raw digital audio files into feature sets that can then be treated in the same way as sets of notes [Typke, 2004].

Common examples of set based models are N-gram models. N-gram models reduce symbolic musical material into N-grammed sequences of intervals, which is then indexed and used in inverted files. [Futrelle & Downie, 2003] It is useful because it narrows the field of potential target records in an indexing scheme, because of its coarse granularity, since N-grams either match queries exactly or not at all.

N-grams works in the following way: N-grams are formed from sequences of intervals. A set of N-grams are then computed for the complete query and for each target by looking at the n pitch intervals and IOI ratios beginning at each successive note. Similarity is then calculated by counting the numbers of N-grams in the query that match the N-grams in the target [Dannenberg & Hu, 2004].

### **2.5.5.2 Hidden Markov Models**

Hidden Markov Models (HMMs) are statistical predictive models to predict the maximum likelihood of a note being present in a frame and then matching frames from possible target and query records. Different features are extracted from each frame and can be used to create training sets. These can include Mell Spaced Frequency cepstral coefficients (which convert multiplicative to additive signals), energy measures and first and second derivatives of the frame that is being investigated [Shih et al., 2003].

Then a selection of features are chosen from the multiple features extracted from the segmented note parts, which would best represent the specific HMMs [Shih et al., 2003].

After the features have been chosen each note is modelled as a HMM. A duration model is then added to account for the differences in the different note lengths. After this the training process starts. First you take a rough estimate of what the note is and then you must decide what the maximum likelihood is for the note, to improve the accuracy of the model. Then you have the recognition process where you encode the note and label its duration. After this the maximum likelihood of a note being the correct note is chosen to represent the note. [Shih et al., 2003]

### **2.5.5.3 Audio Fingerprinting**

Audio fingerprinting is the process of using compact signatures derived from perceptually relevant features to match extracted features from a query against similar stored target information in the database [Cano et al., 2005].

In Audio fingerprinting, fingerprints, are preprocessed to extract sequences of bits of a fixed length described by a feature extractor. [Typke, 2004].

These audio fingerprints are then stored in a database index, along with pointers to the places in the recordings where they occur. The database itself typically consists of inverted lists where a list is held of all audio files whose feature vector contains the corresponding fingerprint. This model differs from the other models because these features do not have to have anything to do with human perception of music on the recording [Typke, 2004]. This makes it fault tolerant to many factors like background noise and bad singers, that other models have problems with.

#### **2.5.5.4 Self-Organising Maps**

Self-Organising Maps are artificial neural network algorithms that are used to cluster similar pieces of music together and classify them [Typke, 2004]. The clusters are ordered in a rectangular two dimensional grid. Information about the clusters is stored within the self organising map neural network as plain ASCII files. Audio files are stored as their corresponding feature vectors. Matching is performed by the network in a nearest neighbour manner [Typke, 2004].

### **2.6 Summary**

In this chapter we reviewed some fundamental concepts that will be used further in the project. We aimed to create a conceptual space from which to view the project in terms of where my project is positioned and how it fits in within the field of information retrieval. This is why a broad overview of storage structures for digital audio data like repositories and musical digital libraries were given. We then moved on to discuss perceptually relevant parts of music that are necessary factors for formulating queries for audio databases. After this we discussed the fundamentals of digital audio data and how sampling, audio file formats and audio formats relate to one another. We then moved on to give summaries of different processing techniques and feature extraction methods that are used in extraction, matching and retrieval.

# Chapter 3

## **3 Frameworks and Toolkits**

### **3.1 Overview**

This Chapter surveys frameworks used in the design of content based audio query systems, projects directly related to music information retrieval and external libraries providing functionality to certain frameworks.

### **3.2 Information Retrieval Frameworks**

#### **3.2.1 CLAM**

CLAM is an object orientated music processing framework developed by Universitat Pompeu Fabra in Spain. CLAM includes components for tasks for managing audio and MIDI devices, signal processing classes and embedding and integrating visualisations from multi platform third party graphical tool kits [Amatriain & Arumi.2005].

The system is organised as processing objects deployed as an interconnected network as can be seen on figure 3.1. Each processing object is able to access processing data tokens and then modify them according to the algorithm that is implemented by the particular processing object. Processing composites are created when different processing objects are interconnected as a network. Flow control schedules guides how these different processing objects, composites and sometimes whole networks interact with each other in the order specified in the flow control schedule. Flow control schedules are executed at runtime [Amatriain, 2004].

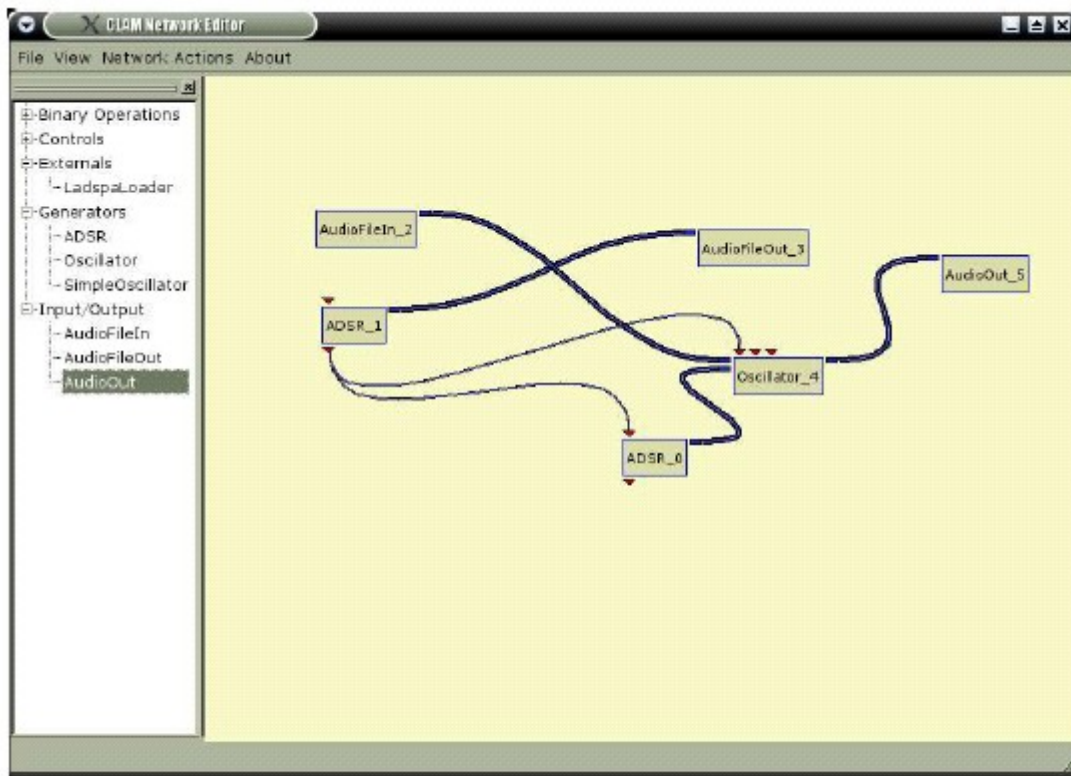


Figure 3.1: Image of a CLAM flow control schedule

There are processing classes for handling both asynchronous data, in which data is fed into the processing class via an open port, and continuous data, where data is from released from controls whenever an event is triggered.

One of CLAM's big strengths is that it contains many complex audio processing algorithms including some for spectral modelling and transformations, feature extraction and classification. It also is platform independent, and can compile on UNIX, MacOS -X and Microsoft Windows platforms.

Although CLAM is object orientated, it exists within the broad CLAM framework. The implications are that processing objects can not operate independently outside the CLAM framework, because objects within the framework are incompatible with any other processing objects from other frameworks, even if the data share the same file format and programming language.

Individual core objects have many dependencies on other objects within the framework. There are also many dependencies on third party libraries. This necessitates the use of an external build tool,

because of the complicated connections between the different objects. This makes it difficult to compile small objects, because of so many other objects have to be included in the composite object.

There is also a visualisation module available in CLAM version 0.7.0 for the Microsoft Windows operating system. Its two uses are to graphically inspect objects and to aid debugging. It was designed so that it can be easily decoupled from the rest of the CLAM framework. The visualisation module infrastructure can be used with other visualisation toolkits like FLTK. [Amatriain,2004]

There is no support for importing modules from other programming languages and different frameworks. [Amatriain,2004] [Amatriain & Arumi,2005].

### **3.2.2 M2K**

M2K is a project initiated by the MTG Group at Indiana University initially intended as an extension and add-in for the D2K Data Mining framework. This is one of the biggest collaborative initiatives between different working groups involved in music information retrieval with various individuals and groups contributing different modules written in Java to extend the framework. [IMIRSEL,2004] [IMIRSEL,2005]

Another purpose of this initiative was to create a framework for MIREX, which is a competition in which different research groups are given the opportunity to test their systems using a set of standardised test queries and results.

D2K itself provides an integrated framework and includes tools for browsing and configuring M2K modules, testing M2K modules and viewing generated visualisations. The system provides an intuitive interface to see and manipulate graphical high level abstractions of modules, with modules changing position by being manipulated through drag and drop functionality.

Both D2K and M2K as well as the core modules are written in Java, making the system platform independent. To build new modules or to configure modules you intend using, a number of parameters must be set for each, including: the command it will run and any parameters that must be passed to it, a working directory to run the command in, either a manually set output filename or an extension to add to the input filename to produce the output filename, and an algorithm calling format String, which will be used to produce the command that will be run on the command line.

Developing M2K applications in D2K then involves assembling processing modules into an itinerary characterising the data flow between modules. Itineraries can then be run as stand-alone



applications on clusters of machines, but you have to write all the itineraries in Java.

[IMIRSEL,2004] [IMIRSEL,2005]

Once an itinerary has been developed, it can be used as a module in any other itinerary, allowing for applications of arbitrary complexity.

There are two types of external integration modules available to import other languages and binaries into the D2K framework. One of the external integration modules is specifically designed to import experiments from Matlab. A general purpose version will run the commands and output the results to the D2K console. The Matlab version has the ability to direct the output to the Matlab console window.

One of the potential problems of the D2K framework, is that it is proprietary software, although it is available freely for academic use. M2K is not totally dependent on the D2K data mining framework. An alternative to using D2K with M2K is using an open source framework called Celerity. Celerity is also written in Java. The only difference is that there is no visualisation module in the Celerity set-up.

This framework would benefit by having an independent operational module, that would almost completely automate the process of annotating and wrapping pre-existing modules from other frameworks for import it into M2K. [IMIRSEL,2004] [IMIRSEL,2005]

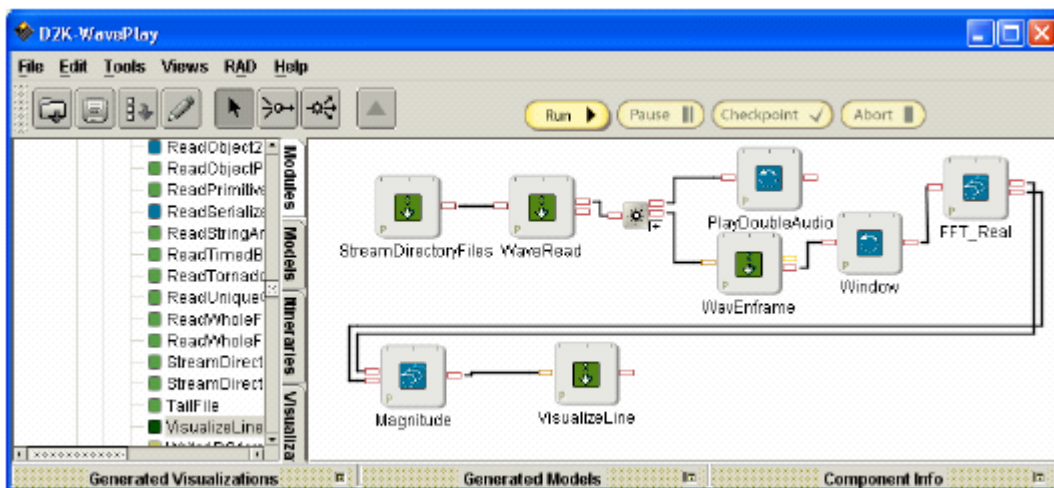


Figure 3.2: Image of M2K workspace with flow control network

### **3.2.3 MARSYAS**

MARSYAS is a framework that allows expert users to build sound analysis and synthesis software. Naive users can interact with the system with high level scripts and graphical user interfaces, while expert users would interact with the framework directly and be able to create new native types and classes by writing and modifying code [Tzanetakis,2001].

Basic modules of the framework are called mar systems and include functionality to implement basic data processing tasks. Mar systems are combined to form data flow networks that are called composites. These composites can be modified and controlled in real time.

The conventions that each mar system must follow are: that each Mar system's main method must support the method in which two arguments (both arrays of floating point numbers) are used to represent slices, each mar system must support the process method that handles the data flow and the update method should handle the control messages. The consequence is that new mar systems are difficult to build from scratch, so one has to extend already existing systems to get a new mar system.

The biggest distraction in interacting with the framework is that individual mar systems have many interdependencies on other mar systems. This makes it difficult to produce small independent mar systems. The same problem is present in the CLAM framework.

This framework is biased towards GNU/Linux distributions, with many features missing from the Windows distribution present in the other one. There is also no native visualisation environment to use with MARSYAS. [Tzanetakis,2001]

MARSYAS is largely independent of external libraries as opposed to CLAM which is very reliant on them. MARSYAS outputs results in .aiff format which is used by the Weka machine learning tool kit. Some of the MARSYAS code was adapted to be used in the M2K toolkit.

### **3.2.4 MUSART**

MUSART ( Music Analysis and Retrieval Technology) is a fully operational audio content based retrieval system. In addition to the basic repository of collections of queries, analysis software and search tools are also included. What makes this project interesting is that it allows different approaches of extraction to be directly compared by allowing a variety of analysis subsystems to

be integrated within a single architecture, allowing objective comparison between different approaches.

The MUSART system includes music retrieval techniques like - Hidden Markov Modelling, fixed frame melodic contour matching with dynamic time warping and a phonetic streams [Birmigham et.al.,2003].

The records are preprocessed using a collection of tools to build abstract representations of the music. The extracted information is then translated to multiple representations. Queries are also translated, and a search engine is used to search the database. It is designed in this way so that various modules and representations can work together, in parallel, or in sequence, to achieve more refined searches.

MUSART uses sung queries from three different groups as target queries. From these queries MUSART automatically builds a thematic index of the pieces in its database. This reduces the amount of data in the source database by only including the themes of the piece of music. Disadvantages of this strategy is that the target queries could match parts of the database that were already excluded by the preprocessing stage [Birmigham et.al.,2003].

The output from the tests include statistical information about the search results. There are different separate databases of target queries, source songs as well as intermediate representations. There is also a separate file for each query that lists all the correct targets. Tests of the search systems are also in a results directory containing text output summary for future analysis.

### **3.3 Music processing languages**

Music composition languages are mostly aimed at music synthesis as opposed to analysis, but it is included here because in the bigger context of broad frameworks these play a big role in the creation of music systems. They sometimes provide processing models for signal processing.

#### **3.3.1 *Matab***

Matlab is a high level language for technical computing and provides an interactive environment for

the development of algorithms, data visualisation, data analysis, numeric computation and building graphical user interfaces [Mathworks, no date].

The main workspace layout in Matlab has various elements. There is the Matlab Editor which provides standard text/code editing and debugging features, the M-Lint Code Checker that Analyses code and recommends changes to improve its performance and maintainability and the Matlab Profiler that records the time spent executing each line of code.

The real advantage of using the Matlab environment for music processing applications is that there are several add-on tool boxes available to Matlab users to extend the environment for music and signal processing. These include the Bayesian tool box that includes many conditional probability distributions and various probability-based algorithms, the auditory toolbox, the netlab toolbox and the SOM (Self Organising Map) toolbox for Matlab.

Functions exist to integrate Matlab based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel. Matlab code can also be called from C and Fortran using the Matlab engine library. Other frameworks also make an effort to accommodate Matlab users by allowing Matlab to be used within them.

The big problem with Matlab is that the framework is proprietary, which means that add-ins to the language cannot be redistributed. An example of the Matlab workspace can be seen in Figure 3.3.

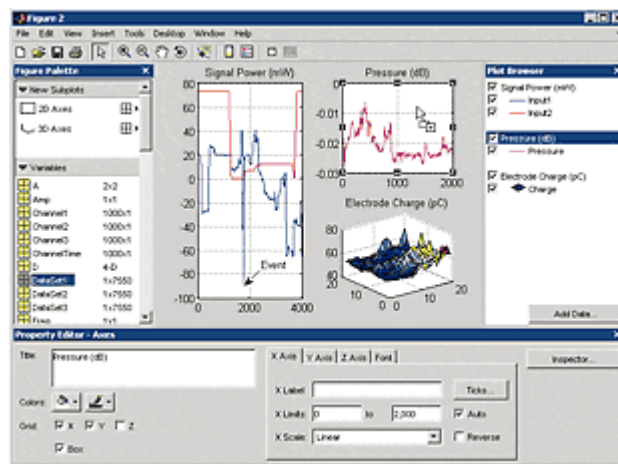


Figure 3.3: Image of the MATLAB workspace

### 3.3.2 Octave

Octave is a high level interactive language. It provides a framework that is comparable with Matlab and caters for numerical modelling and graphic visualisation of musical data. Matlab programs can be ported into Octave.

Octave can do arithmetic for real and complex scalars and matrices, solve sets of non-linear algebraic equations, integrate functions over finite and infinite intervals, and integrate systems of ordinary differential and differential-algebraic equations [Eaton,1998].

Octave uses the GNU readline library to handle the reading and editing of input. Two and three dimensional plotting is fully supported using gnuplot. The underlying numerical computations are done using standard Fortran packaged in a library of C++ classes. If possible, the Fortran subroutines are compiled with the system's Fortran compiler, and called directly from the C++ functions. For this reason octave is not that platform independent and operates only on UNIX like systems and requires the GNU C++ compiler. An example of the Octave environment can be seen in Figure 3.4.

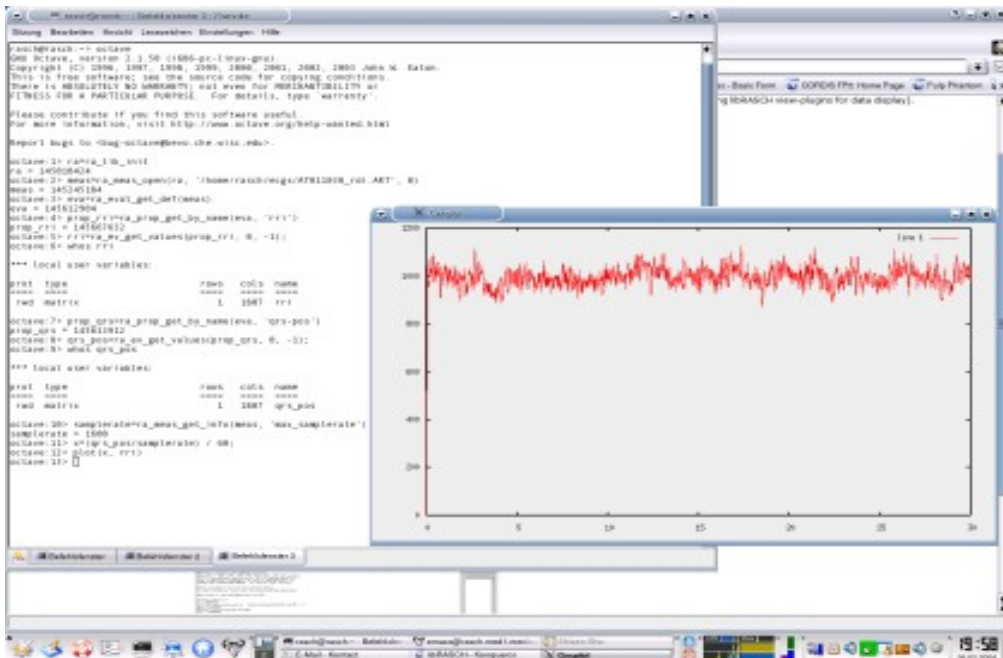


Figure 3.4: Image of the Octave environment

### 3.3.3 Labwindows

Labwindows is a C programming and development environment which mostly deals with developing measurement applications. It includes a large set of run-time libraries for instrument control, data acquisition and analysis. It includes tool kits for digital signal processing, but also UI design, data analysis and visualisation, built-in instrumentation libraries (GP IS, DAQ, analysis) and Instrumentation-based user interface controls (graphs, knobs) [National Instruments Corporation,2006].

An example of the Labwindows environment can be seen in Figure 3.5.

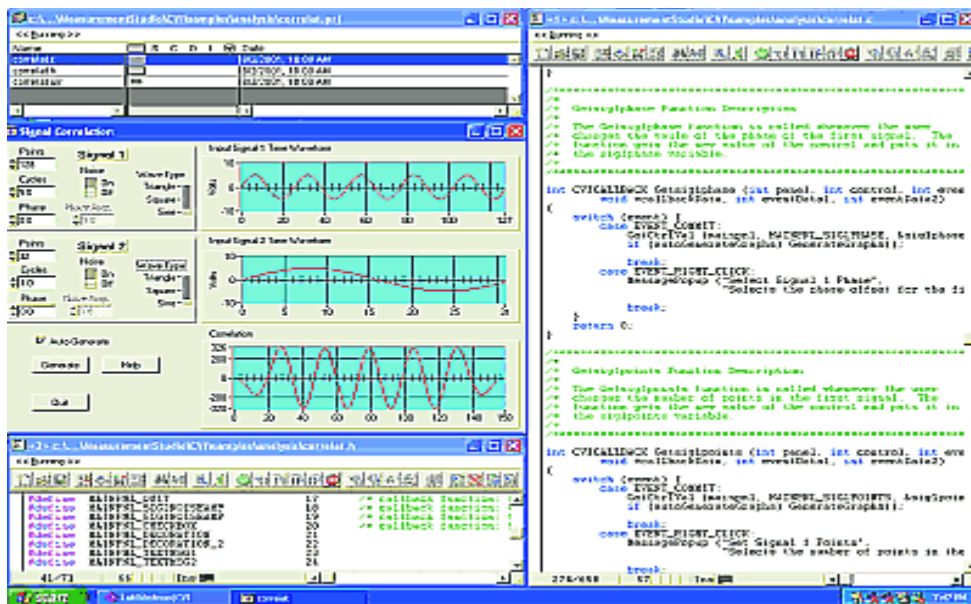


Figure 3.5: Image of the Labwindows environment

### 3.3.4 Nyquist

Nyquist is an open-source language environment for sound analysis and synthesis. It is implemented in C and C++ and runs on Win32, OS X and Linux.

Nyquist offers a powerful and efficient functional programming model for signal processing, and is particularly good at working with large amounts of data because it automatically streams data rather than allocating large arrays in primary memory [Lamere,2005]. In addition to audio processing, Nyquist offers a full Lisp interpreter, with which you can create your own custom signal processing classes using Xlisp, an object orientated subset of the LISP language.

## **3.4 External Libraries**

External libraries are important because provide key functionality for frameworks to handle music signal processing, machine learning and visualisation, so that there is no need to replicate processes and classes within the framework that have been implemented very efficiently somewhere else or do not form part of its core functional classes.

Interaction is uniform with classes that look like any other class in the framework, the interaction details with the actual library being taken care of by the framework. But you also have the problem of having different library distributions for different platforms. To distribute a program written in the framework, you have to include all the libraries or DLL's that are used indirectly. These are mostly standard and everyone uses them.

There are many varied libraries that offer support for frameworks that are written in Matlab and C++. There are a lot less libraries available for languages like JAVA. Many of the libraries that are available, have support is skewed towards Unix/Linux platforms.

Many of the frameworks have optional or core dependencies on other projects and external libraries that are necessary for the frameworks to function.

### ***3.4.1 Machine learning libraries***

Machine learning libraries range from ones that train different algorithms to neural optimisation development, to ones that solve various regression and classification problems. The libraries also support many types of conditional probability distributions, decision nodes, utility nodes, chance nodes and many different inference algorithms, pattern recognition and implementations of several popular auditory models [Lamere, 2005].

### ***3.4.2 Music processing libraries***

The core function of music processing libraries provide methods for controlling *I/O* of audio data and enable basic MIDI input and output classes. Other functions are to simplify interaction with computer audio hardware and to allow sound samples to be accessed though standard library interfaces.

### **3.4.3 Visualisation**

There are three uses for visualisation toolkit libraries in frameworks; 1) To provide functionality for plotting graphs and representing other numerical data 2) To create custom interfaces and provide tools from within frameworks to create user interfaces for applications and 3) To create visualisation environments for the framework itself while allowing users to interact with the underlying classes at a higher abstraction level.

Examples of these are the qt library that provides a complete application development framework for creating applications using C++ , and the MFC library that provides a collection of classes that can be used in building application programs. The wxWidgets class library allows the compilation of graphical C++ programs on a range of different platforms, by defining a common API across platforms that uses the native graphical user interfaces on each platform [Lamere,2005].

## **3.5 Other Tools**

### **3.5.1 Sphinx-3**

Sphinx-3 is a speech recognition system that is used by MIR researchers to calculate mell spaced cepstral coefficients. The S3 decoder is based in the Viterbi search algorithm. Its input is pre-recorded audio specifically pre-recoded speech. This is done by the front end of the module. Only the acoustic model is used by MIR researchers [Seltzer, 2002][Ravishankar,2004].

### **3.5.2 WEKA**

WEKA is a machine learning system written in Java, initiated by the University of Waikato in New Zealand and stands for Waikato Environment for Knowledge Analysis. It provides implementations of learning algorithms and includes tools for transforming datasets. The main functions of the WEKA environment are to feed datasets into a learning scheme and to analyse the resulting classifiers and to extract information from the resultant data. It also allows users to access the libraries from their own Java programs in order to write their own machine learning algorithms. Several learning schemes can also be applied and their performance compared. All the learning schemes have the same command-line interface and they are all measured by a common



evaluation module [Witten & Frank, 2000].

WEKA is a non real time system and only takes text input and only accepts CSV and .arff data files. WEKA modules has to be written in Java or use the data structures and transforms/filters available in WEKA [Witten & Frank, 2000].

### **3.6 Summary**

This chapter gave an overview of different tools and frameworks that create content based MIR systems. This chapter documents a few attempts at the creation of test-beds and testing architectures for both event based (e.g. MIDI) and audio content based MIR to show in which ways interoperability and direct testing has been achieved. As this is the foundations from which the interface was built. This chapter also gives an overview of tools, libraries and music processing languages available so that the complexities of testing different components created with different tools can be fully appreciated.

# Chapter 4

## **4 The MIRMaid Interface**

### **4.1 Overview**

This section presents an interface for a test-bed called MIRMaid, an acronym for Music Information Retrieval Modular aid. The interface allows different combinations of parameters to be combined and tested against each other to determine an optimal set of parameters for different problem domains in content based audio retrieval systems.

The repository of the test-bed contains different versions of each audio file, which are processed by different transformations. Different transformations are applied to the data through independent modules that are imported from other music processing, data mining or signal processing frameworks. These transformations can then be combined with the help of the interface into different combinations, in which one's performance can be directly compared against another.

### **4.2 Structure**

#### **4.2.1 Test-bed Structure**

The test-bed consists of three different elements: the repository that houses two collections of audio data files, each collection containing different versions of the same files that have been processed by different transformations; the processing transformation modules that transform audio records; and the interface, which allows users to link different modules in any order and execute queries and query comparisons.

#### **4.2.2 The repositories**

There are two kinds of audio data objects that can be distinguished between in the repository: The first is the audio/sound clip, which is the original piece of music that has not undergone any type of

transformation; and feature vector, which is a sound clip that has undergone one or more transformations.

Each data object, regardless of whether it is a feature vector or a sound clip, would be encapsulated in an interchange format. The metadata will provide both high and low level data on file type, file format and other important information like behaviour and content. This would be done to make sure that all data objects' information can be easily and consistently accessed and to make it easier for data to be incorporated into larger test-beds.

Each sound clip in the repository will have a set of corresponding feature vectors. Each feature vector would have undergone processing by a specific transformation module or module network. Each time a new module is added by the user to the test-bed, a new feature vector set is created for each collection in the repository.

This speeds up the comparison process, as it minimises the execution time by decreasing the number of feature vectors or sound clips needing transformation at each stage of the process.

### **4.2.3 Modules**

Modules represent the smallest computational units in the test-bed. They are little programs that represent transformations of the data in the repository.

All modules are in the form of binary programs, and specify the format of the data they want imported and the format that the data is in after being processed. They are created by external frameworks in any language and packaged in binary format, before they are imported into the test-bed. The import process relies on the user to give correct information about the input and output format of the modules they are importing into the test-bed. There are two big requirements for the modules that could be imported into the test-bed: the input and output formats for the module have to be specified and modules have to be able to be executed on their own and not be dependent on other external libraries or classes.

The modules can either be executed alone or they could be linked as a network of objects, that can be called module composites. Modules can be ordered in the network in any order provided that the input format of the current module corresponds with the output format of the previous module. Both modules and module composites will have corresponding feature vectors.

The repository and the modules will further be expanded on in the Future Work Chapter. The rest of this chapter and the next two chapters will be dedicated to the discussion on the development of the interface.

## **4.3 The Interface**

### **4.3.1 Design Goals**

We set out three primary design goals for the interface.

The interface had to be simple enough to be used by novice users who only have basic knowledge of music but who are computer proficient. Even though this group of users has no prior exposure to MIR frameworks and test-beds, making the interface simple enough to be used by novice users would ensure that the interface will be intuitive enough for expert and specialist users.

The interface had to be usable. If the interface is usable it would ensure that the interface is practically applicable to MIR needs and could actually be used in real world situations if the rest of the test-bed was implemented. It should include functionality that complements MIR analysis tools and test-beds that expert and specialist use.

The interface had to be adequate. It should be designed so that it would perform adequately in common tasks associated with using the test-bed.

### 4.3.2 Interface Elements

The interface consists of different interface elements. Each element correlates to a discrete task executed by the user to execute a query.

#### 4.3.2.1 The "Choose a repository" component

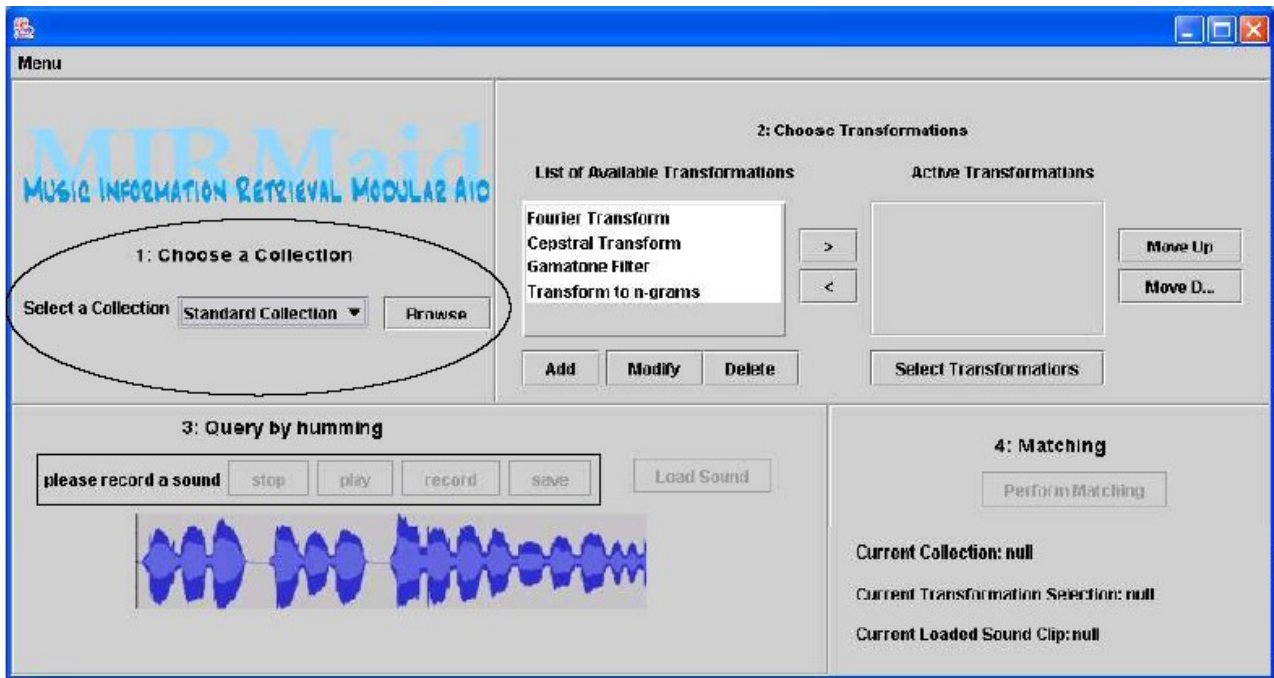


Figure 4. 1 : Image of the "Choose a repository" element within the interface

In the "Choose a repository" component (Figure 4.1). potential users can choose a collection of music from the repository or specify a custom repository on the system or network which they would like to use in a query or comparison.

### 4.3.2.2 The "Select Transformations" component

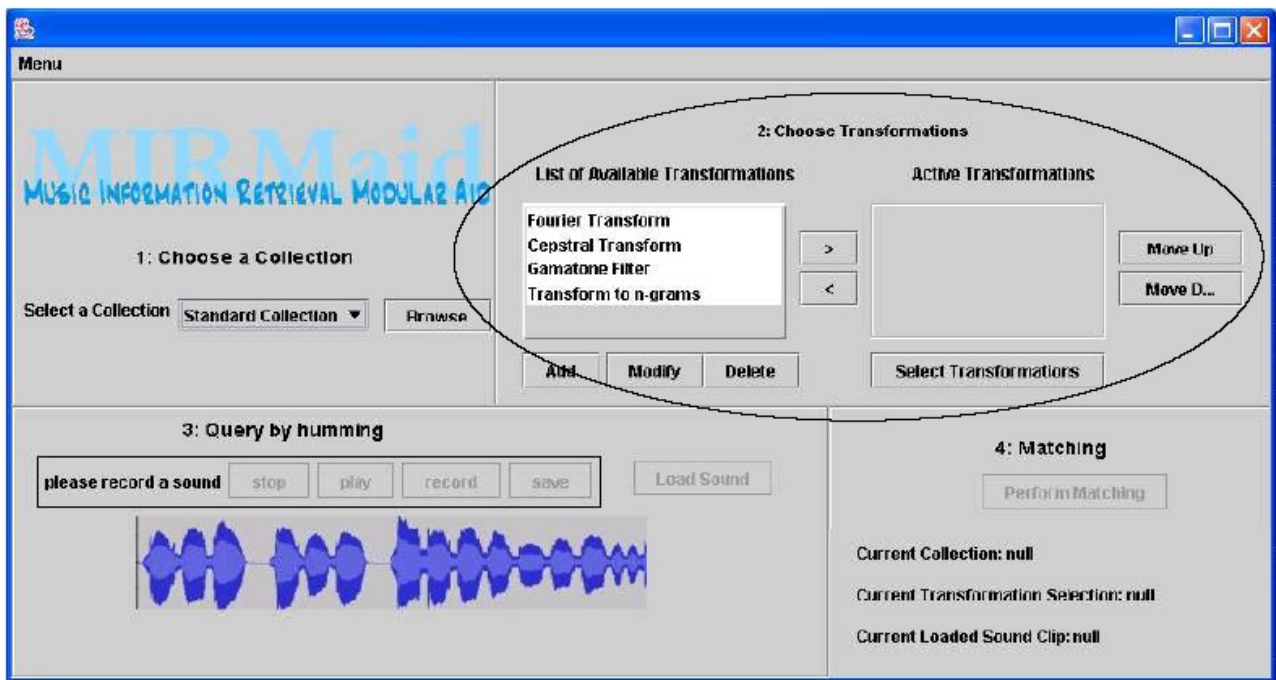


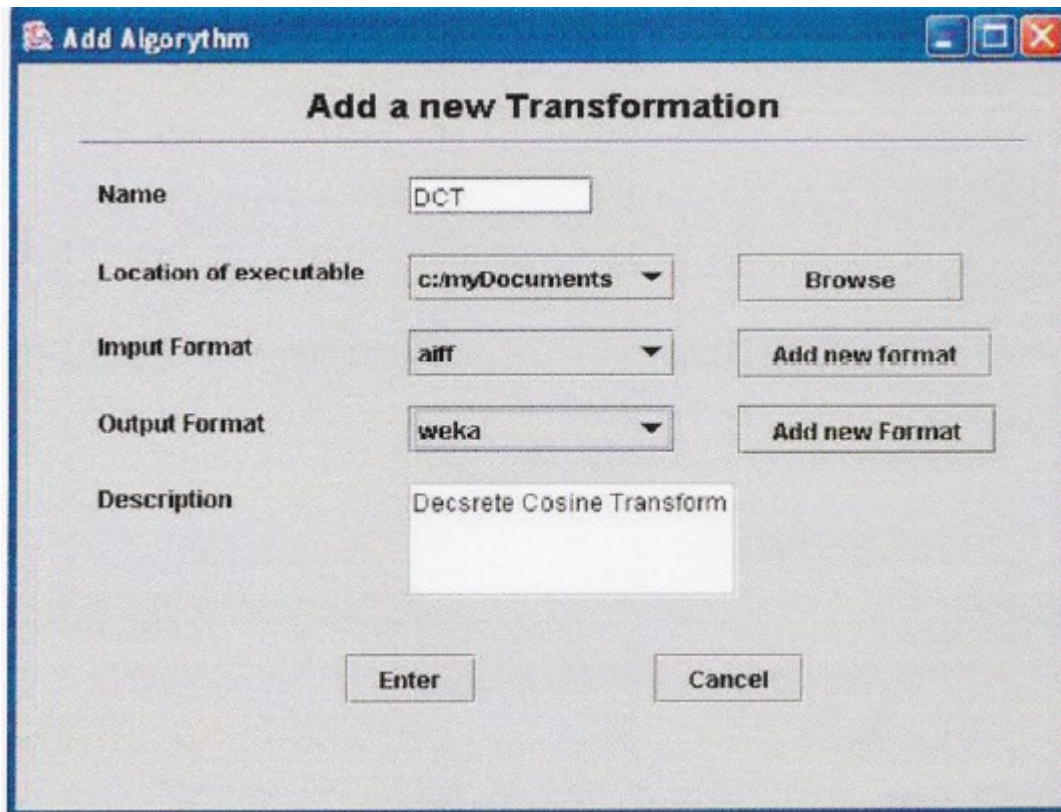
Figure 4.2 : Image of the choosing transformations element within the interface

The choose transformations component (Figure 4.2) allows transformation modules to be selected, combined and ordered into module networks, that are then applied to the chosen collections in the repository. This component allows transformations to be ordered by either pressing the "Move Up" or "Move Down" buttons in a desired order.

This component also has to return error messages. If one of the modules selected is not compatible with the rest of the modules chosen, in the desired order, this interface component should show an error message saying that the selected module network combination was unable to be executed.

If there are desired transformations that are not present in the list they can be added through the transformation window, by pressing the "Add" button below the "List of Available transformations" list.

### 4.3.2.3 Adding Transformations Frame



The image shows a Windows-style dialog box titled "Add Algorithm" with a sub-title "Add a new Transformation". The dialog contains the following fields and controls:

- Name:** A text input field containing "DCT".
- Location of executable:** A dropdown menu showing "c:/myDocuments" and a "Browse" button.
- Input Format:** A dropdown menu showing "aiff" and an "Add new format" button.
- Output Format:** A dropdown menu showing "weka" and an "Add new Format" button.
- Description:** A text area containing "Decrete Cosine Transform".
- At the bottom, there are "Enter" and "Cancel" buttons.

*Figure 4.3: Image of the adding transformation element*

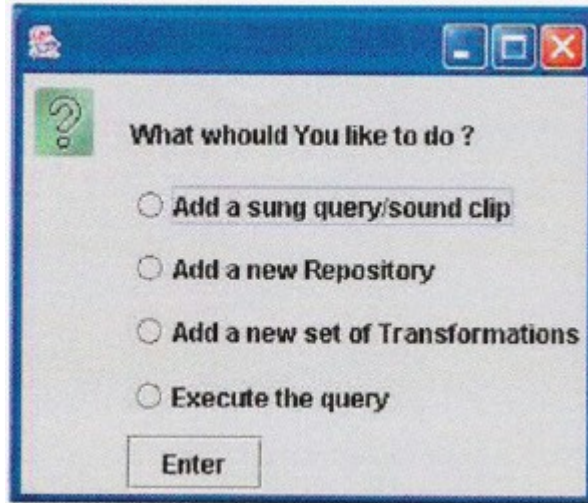
New transformations are added through the transformations form (Figure 4.3) after the "Load Transformations" button is pressed on the interface. This form should support standard input and output formats used by other frameworks. If there is no suitable format users will be able to specify their own custom format.

With some output and input formats there are many complicated parameters that need to be specified. Many novice users would only know the basic parameters of the format and would have to guess the rest so that they can continue with the query. This problem was solved by using default values for users who do not specify any extra information even if it is required as described in [Tidwell, 2005]. This functionality was not necessary for simpler and better defined formats.

There is a fine equilibrium in the trade-off between requiring too much information from the user and getting enough information so that processing modules can be correctly executed. Requiring too much information induces users to guess the parameters that they do not know to continue with the query, causing transcription errors. Requiring too little information restricts the flexibility of the modules. Some implications are that if there are any errors the information specified by users

the one format may be seen as two different formats. There would also be incorrect conversions that may end up corrupting many of the correct annotations and queries that use the corrupt annotations. There would be false incompatibility between transformations that will be processed consecutively. Therefore drop-down boxes and buttons were used in order to add new file formats.

#### 4.3.2.4 Use case Control component



*Figure 4.4 : Image of the control form*

The use case control form (Figure 4.4) allows the user to navigate back to areas that they have already visited but need to visit again, in order to add multiple repositories or groups of transformations to preform comparative queries. All queries pass through this component after users have selected transformations. The component is activated through the "Select Transformation" button on the interface. The simplest way to implement this component was to use radio buttons on a pop-up screen.



### 4.3.2.5 Sound loading/recording component

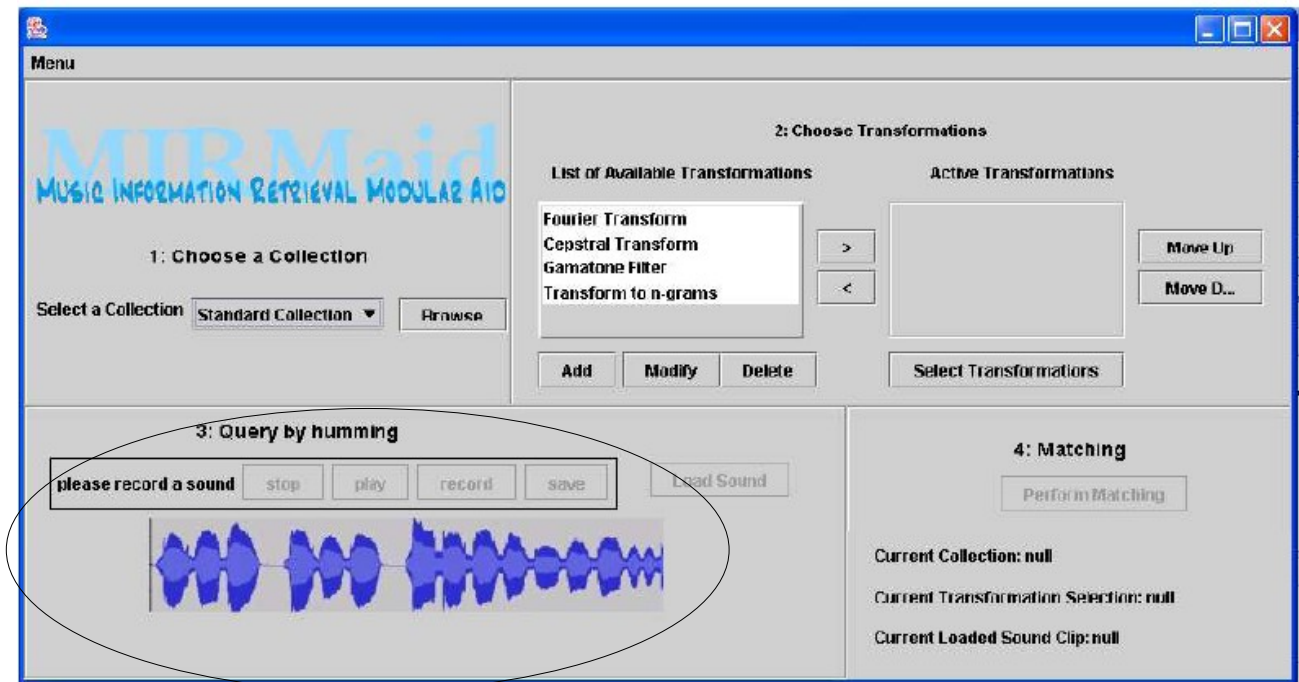


Figure 4.5: Image of the sound recording element within the interface

This sound loading element (Figure 4.5) allows users to either add a new query by humming into a microphone or choose a sound clip from a sound clip collection, in order to do a known item spot query on a repository after user selected transformations has been applied to the data in the repository. This component also allows a query to be saved and later loaded from memory, so that the query does not need to be entered every time you want to repeat the same query with other parameters or a different repository.

### 4.3.2.6 Matching/Evaluate/Execute the query

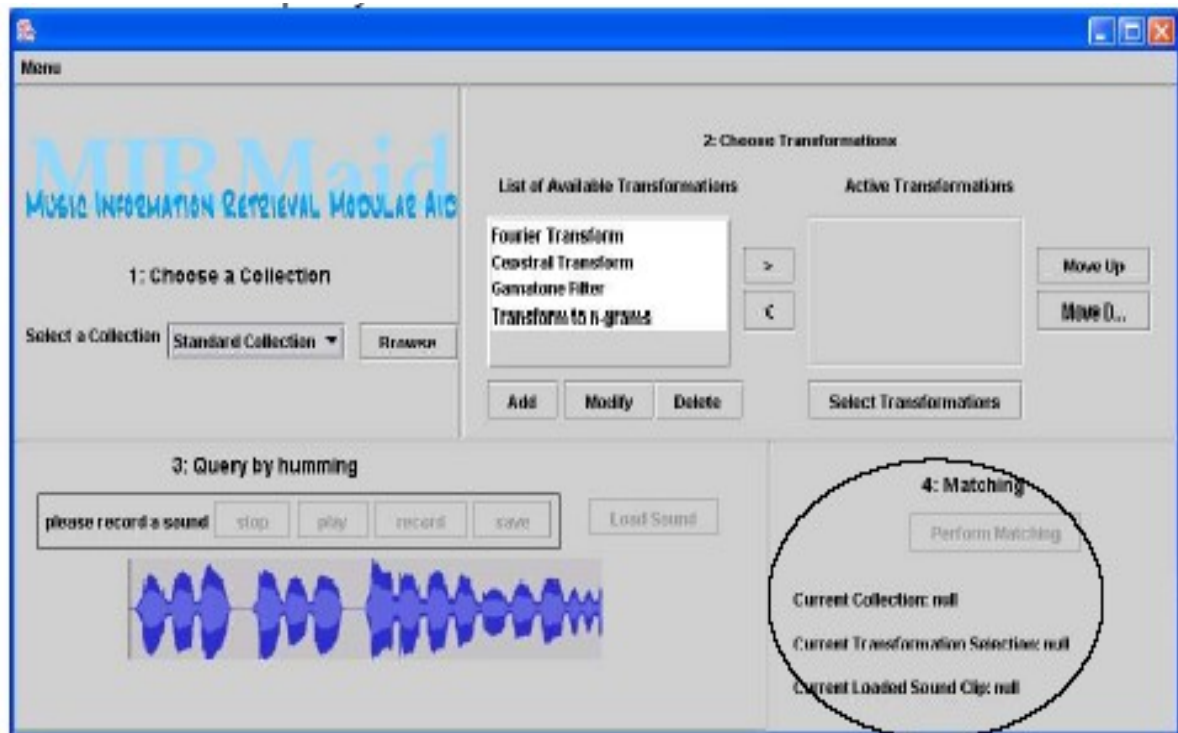


Figure 4.6 : Image of the matching element within the interface

The "matching" element (Figure 4.6) essentially consists of one button. The button causes either a comparative query or a known item query to be executed.

Update and Status areas give feedback on where users are in the query process and what parameters they have chosen. Update areas also provide prompts for the current task they are doing.

### 4.3.2.7 Presentation of Results

Results are presented to the user in two different ways. The results are either returned as a set of collated statistics representing the comparison that was required by the query, or a set of records in a graph returned by a spot query.

The two different forms that present the data have slightly different formats for presenting the data in. The first form (Figure 4.6) presents the results from spot queries as a ranked list in order of relevance. This table also allow you to press buttons so that you can start and stop playback of

sound clips.



Figure 4.6: Image of the results screen.

The second frame (Figure 4.7) presents the results from the comparative query back as a graphic, on the performance of the compared variables of the query. This element also included a button allowing users to start a new analysis.

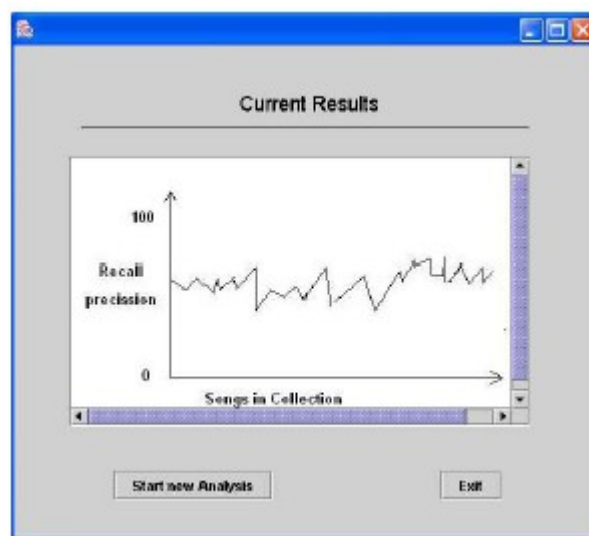


Figure 4.7: Image of the second results screen

### **4.3.3 Interface development**

We used an evolutionary prototype approach when we built the interface. This consisted of a combined analysis and requirements specification stage, after which we implemented a basic prototype interface and held informal interviews in which the first iteration of the interface was evaluated and modified. After experimentation the interface was modified again to reflect the changes and feedback from the experimentation.

The initial analysis stage and requirements specification stage took the form of an extensive review to figure out where exactly the interface would be positioned for the test-bed. We did informal interviews with users who had expert knowledge of music, signal processing and computers, but were not involved in music information retrieval. These informal interviews were to verify the feasibility of the interface.

### **4.3.4 Strategic positioning of the framework**

One gap we identified in other test-beds was that there was little support for inclusion of small specialised repositories, like ones containing African traditional music. The other problem we identified was that better tools were needed to combine modules built in different frameworks, so we decided to position my interface in a way that it would satisfy the needs of users who want to import pre-built modules quickly and easily and test-bed users who wish to perform an experiment. Most of the lower level details, like formatting data or tuning modules to interact with each other are hidden from the users unless they specifically request control over lower level details.

### **4.3.5 Interviews**

We conducted both individual and group interviews. All the interviews were done with the help of an initial interface prototype (Figure 4.8). This was done to facilitate discussion on the interface and to give a concrete visual representation of the prospective system for both expert and novice users. This interface was basic and only contained broad elements and vague representations of the elements that would eventually be represented in the interface.

The reasons for the interviews was to determine the best placement of the various interface elements and get a perspective on what users that closely resemble future users of the interface wanted in order for them to perform queries using the interface.

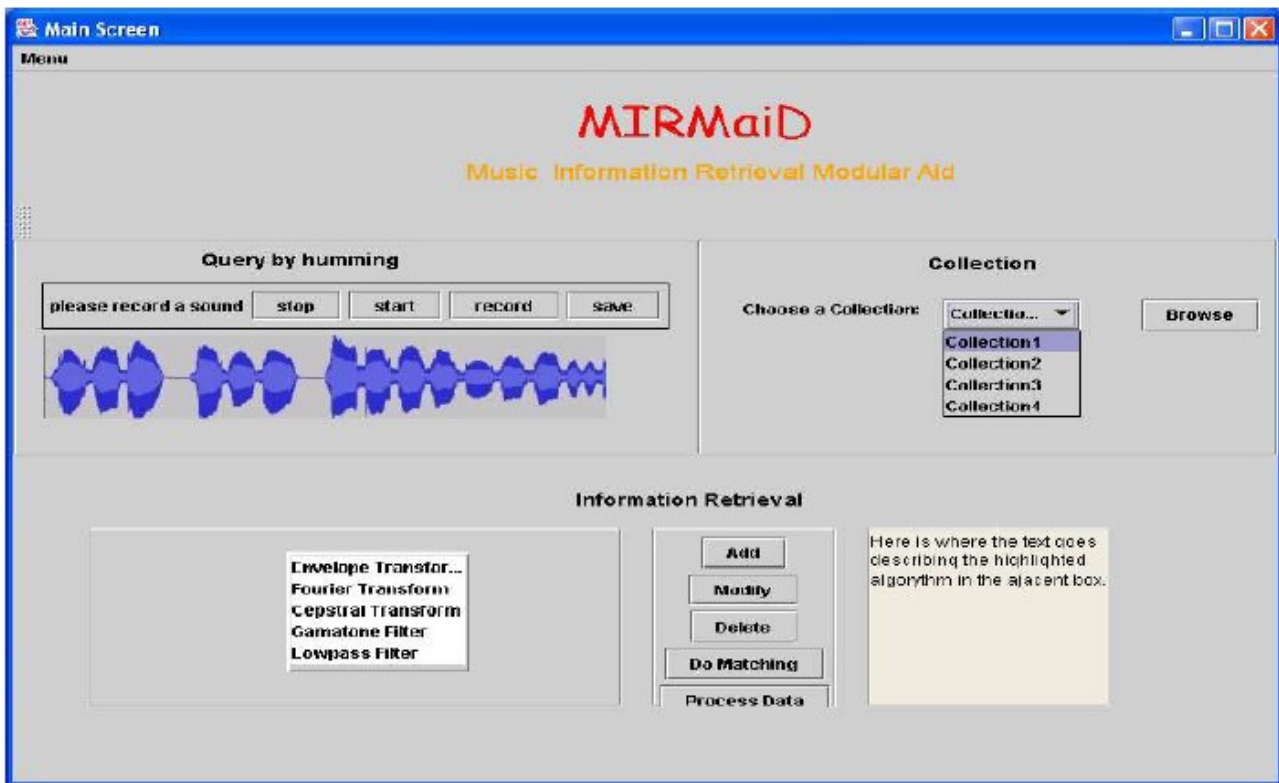


Figure 4.8: Image of the main screen of the first interface

The group interviews were in the form of an informal discussion in which the scope of the study was first explained, what exactly my interface planned to do and different aspects of the proposed interface. Then a walk-through of the interface was done and we discussed attributes like navigation, layout and browsing.

Four subjects were interviewed. They were all involved in research in Computer Music and had expert knowledge of music, and were very proficient in using computers for music sequencing and analysis. Two of the subjects we interviewed had expert knowledge in audio retrieval methods and the other two subjects understood all the principles involved in audio retrieval at a more general level.

There was a clear split in opinion at the end of the discussion between the interview subjects in which direction they saw the interface developing. The non-specialist expert users (users with only a basic understanding of the tasks involved in audio-based retrieval, but who are experts in terms of signal processing and other music related disciplines) argued for the process to be as simple as possible and shielding them from the underlying complexity of the modules.

The specialist expert user required more transparency of the system both in terms of information

on the modules that have already been imported into the system and customising modules in the framework itself. The opinion expressed by the expert user was also that the interface would benefit from more transparent interaction within the individual modules.

One of the expert users suggested that visualisation could be added for presenting the results on the results form by using Matlab visualisation tools. A gap was also identified in processing and returning results from queries.

The interface walk-through showed up many problems, things to watch out for and future improvements to the interface.

The first possible improvement highlighted by the walk-through was program flow. There was no clear way in which users knew what navigation path to follow to execute a query and in which sequence they should perform the tasks.

The second possible improvement highlighted was the need for proper error trapping. There were instances during the walk-through where users could change key variables and processes midway through specifying query parameters that could impact directly on the results returned from the query.

The third possible improvement highlighted was decreasing the number of hops in navigation where users had to go backwards and forwards and even hop over some elements to execute a query.

The user interface also had way too much white space, with all the areas not clearly enough defined and without any clear grouping of elements.

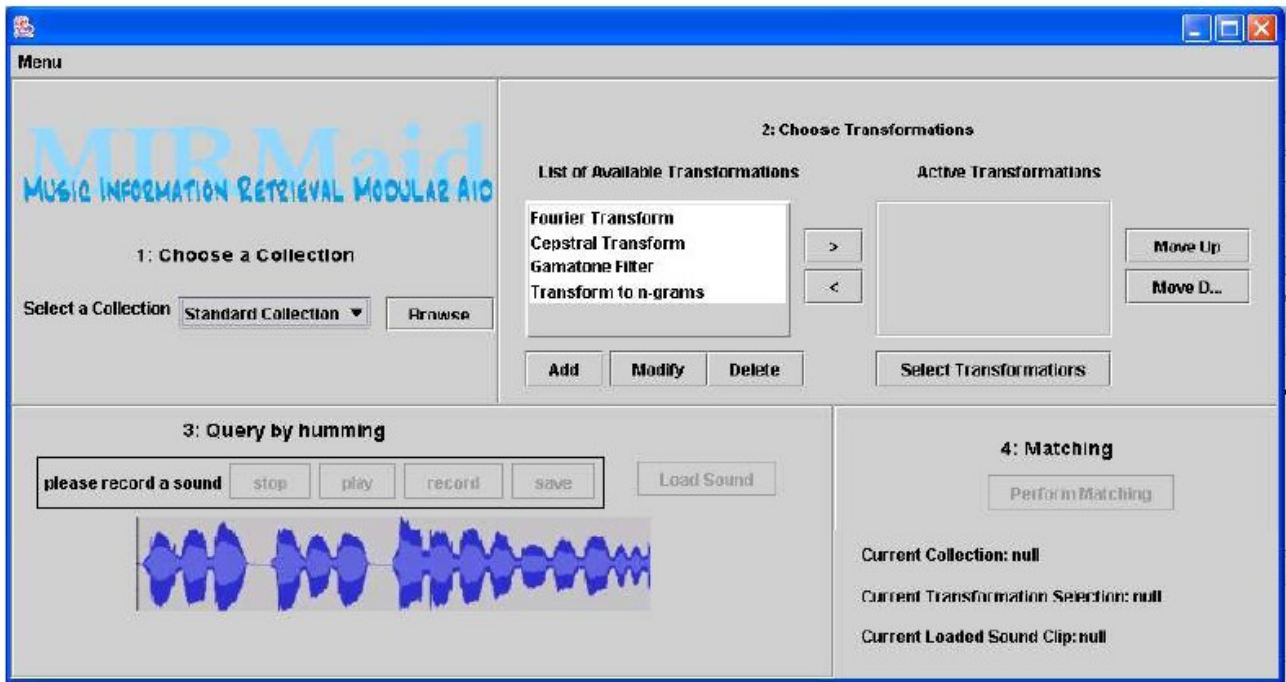


Figure 4.9: Picture of the main screen of the interface after the interview

#### 4.3.6 Improvements on the interface

After the interviews were completed the interface was modified as is shown in Figure 4.9.

One of the big changes to the interface was that order was created to the navigation by adding numbers to the different panels of the interface. This was to allow users to follow stepwise to execute a query.

Elements were reordered in the interface to improve the program flow so that each element would flow naturally and logically into another with the least number of steps to complete an average query using the interface.

An extra layer of guidance was added to the query process through progressive enabling. This ensured that users would not be able to corrupt the query process or the parameters that were already specified.

The space on the interface was more effectively distributed by adding more elements and options for users. Different areas of the interface were also more clearly defined by creating borders around separate interface elements to create a cohesive grouping of elements representing one

distinct step in the query process.

Predictive defaults were placed in editable slots to make it easier for potential users of the interface to complete the most commonly requested queries.

#### **4.4 Description of development tools used.**

Development tools used were Visual Studio. NET run on a computer with Windows XP, the Eclipse and Netbeans IDE's using the Java SDK 1. 4 platform.

#### **4.5 Summary**

This chapter started of explaining different components that make up the interface and how they relate to each other. After that there was a discussion on how the interface was developed and the interview process that helped improve the layout and structure of the interface.

The design of the evaluation system for the interface will be discussed in the next chapter.



# Chapter 5

## *5 Experiment and Questionnaire Design*

### **5.1 Overview**

This chapter explains the experiment and questionnaire design for the user experiments to evaluate the interface. The evaluation strategy chosen for testing the interface was a combination of a user experiments, a questionnaire and user observation to get an overall assessment of aspects of the interface design.

The evaluation was aimed at answering two questions: whether the interfaces is user friendly and if the interface is intuitive. This was done primarily through a questionnaire, which was filled out after the experiment was completed. Test subjects were monitored throughout the experiment to see if they have any problems or had any questions either on the tasks or on aspects of the interface.

### **5.2 Population Selection**

The population sample included both novice users and expert users. The novice users interacted with the interface for the first time while some of the expert users have already been exposed to the interface in previous iterations of its design.

A big concern in the constitution of the overall sample population was that even amongst expert users there are only a few people who would normally find the interface relevant to the activities they are involved in. We aimed to get as many expert users as possible, but that was a challenge since not many people involved in computer music were able to take part in the experiment. We relaxed the requirements for classification of expert users since we were never going to get a true reflection of the population. We decided to supplement the few expert users with a lot more novice users, since the interface was as simple as possible to cater to the needs of novice users.

The experts group can again be subdivided into two groups - a group who are not involved in the

task of musical data mining/information retrieval itself and a group whose expertise lies more in music creation and musicology. The main requirement for classification as an expert user is that the subjects have to have expert knowledge of computer music or at least music.

The requirements for classification in the novice user group was less rigid. The minimum requirement for subjects were that they had to be computer literate. The same tasks were set for expert and novice users.

### **5.3 Experiment Design**

The experiment tested how effective the layout and overall logical structure of the interface was. The experiment was to confirm that all of the sample set of users were able to perform the task that they were set successfully.

Each experiment was done once by all the participants in the interval of three days, in which users were allocated a slot in which they performed the experiment and filled out the questionnaire. Volunteers signed up for a slot beforehand. For the duration of the experiment a room was allocated specifically for user tests. Two days were allocated to expert users and three days allocated to novice users. Testers did the experiment in individual slots. Subjects were not allowed to observe the previous tester doing the task, and a ten minute interval between subjects was arranged.

The order of the experiment was as follows: 1) The consent form was signed. 2) the user did the experiment, 3) the user completed the questionnaire, and 4) users were allowed to give comments on both the questionnaire and the interface in general. The assumption about the population performing the experiments were that they would have basic computer proficiency. This was a valid assumption since the population consisted exclusively of Music Technology and Computer Science masters students.

### **5.4 The Tasks**

The tasks were selected in a way that would make the most sense to novice users of the interface. Users were given broad guidelines on how to perform the tasks and encouraged to ask questions if they had any problem with the tasks they would perform. Copies of the tasks that were given to the

users can be found in Appendix A2 and Appendix A3. Below a summary of the tasks are given.

In the first task the participant was asked to query one test-bed by using a sound clip from Vivaldi's Four Seasons, located in a folder on the local disk drive of the computer the experiment was performed on, for finding the list of records in the test-bed that satisfied the parameters that had been specified beforehand.

In the second task the participant was asked to execute a query that returns a graph that shows the overall performance of selected parameters in matching given a particular music collection.

## **5.5 Arranging outcomes from the interface**

Despite the fact that the interface was only an prototype, we needed to get a more realistic idea how real users would react to the interface. We arranged the outcomes from the interface using a modified version of the Wizard of Oz technique [Web source, 2006] where humans simulate the response by a system and the users are unaware that the system is not real. Typically the Wizard observes the actions of the user from another room and manipulates the responses to user actions in real time.

The user experiments were arranged beforehand to reflect how the system would respond to actions that are performed through the interface. Each task modified so that it would respond in a realistic manner to the actions initiated by the user.

## **5.6 Questionnaire Design**

The purpose of the questionnaire was to find out if the interface is successful in being both user friendly and useful. The information we extracted from the questionnaire was: the profile of the test subject, the classification of users (as an expert user or as a novice user); information on how users rate the task flow of the interfaces, their overall impressions of the interfaces and if there were specific areas in the interfaces that were ambiguous or if they at any point had a problem performing the tasks.

The questionnaire was administered after users completed the tasks set for them. One of the factors considered when deciding on the length of the questionnaire was the cumulative time it

would take the users to perform both tasks.

The questionnaire had two parts: the first part asked profile questions to test subjects and the second part dealt with interface and task related statements.

### **5.6.1 Subject profile**

The reasoning behind the subject profile questions was to find out how proficient experimenters were with using computers and with music manipulation software. We avoided asking these types of questions directly, since people either tend to over estimate or underestimate their experience levels.

From the questions we did ask we were able to group subjects into different categories. The first group were expert users who have both knowledge of music processing and a high proficiency in using computers and music manipulation software. The second group were novice users who either have basic or no knowledge of music. The third group consisted of users who were musicians/musicologists, without a high proficiency in using computers.

### **5.6.2 Interface and task based questions**

The main purpose of these questions was to find out if the overall flow of the workspace was intuitive, if there were any elements overlooked in the previous iterations of the interface design and if the interface performed effectively in the tasks set for the experimenters.

We tried to limit ourselves to a maximum of ten questions, because near the end of a long questionnaire, fatigue sets in and people start replying to questions at random or leaving spaces blank and ignore the more informative open ended questions at the end of the questionnaire. This part of the questionnaire was structured as a set of statements in which experimenters were asked to judge statements on a non-numeric scale of five different alternative ratings ranging from positive to negative.

The first set of questions was concerned with the task itself, because task recall degrades as soon as their focus shifts to something else. On interface based questions experimenters are always able to go back to the interface to help them answering questions. The main objective of the first

set of questions was to assess the perceived difficulty of the tasks to the user. From these responses it is also possible to get feedback on issues in the interface that was not directly asked.

The second set of questions dealt with interface related questions and focused mostly on usability.

A copy of the questionnaire can be found in the appendices as Appendix A4.

## **5.7 Justification for not using time measurement as an evaluation tool**

Task based and subjective measurement of the interface allows for sufficient measurement of the interface. When you get to a more specific level time based evaluation will make more sense.

Measuring time on a prototype instead of an operational production system would ignore time based variables like, loading time, processor speed and program execution speed.

## **5.8 User observation**

Only one user performed the experiments in one session. This allowed easy observation of test subjects to: 1) help whenever there was a problem; 2) observe the order in which experimenters executed the steps of each task and 3) control any critical situations that might have impacted on the execution of the experiment.

The observational technique that was used was requiring users to think - aloud, unless it interfered with their ability to do the experiments. This was one of the main reasons why the tasks were not timed.

After the questionnaire was completed there was a short post-task walk through if the experimenter did not think aloud while performing the tasks, to discuss alternative task executions that were not pursued by the user and reflect back on the actions in a more robust and meaningful way than would be possible through the questionnaire.

## **5.9 Confounding variables**

A small pilot study was done ahead of the main user experiments to determine any obvious usability or technical problems to reduce confounding factors on the experiments.

The pilot study was executed on a mobile computer without a mouse attached. The absence of a mouse had a big impact on the time it took to complete the user experiments. As a result of this information, a mouse was added to the computer setup for the experiments as this would have been a confounding factor in perceived difficulty when dealing with the interface even when timing the execution of the experiments was not an issue.

The other confounding factor that had an impact on execution time was excessive background noise. This was controlled by isolating users from other people by doing the experiments individually in a closed room.

## **5.10 Summary**

This chapter dealt with how the user experiment was set up. Firstly the population selection was discussed and how it will influence experiment design. Then the design of the experiments were explained and purpose behind them. Afterwards the questionnaire was discussed and the reasons behind some of the questions in the questionnaire. Lastly confounding variables were discussed that might have had an impact on the study.

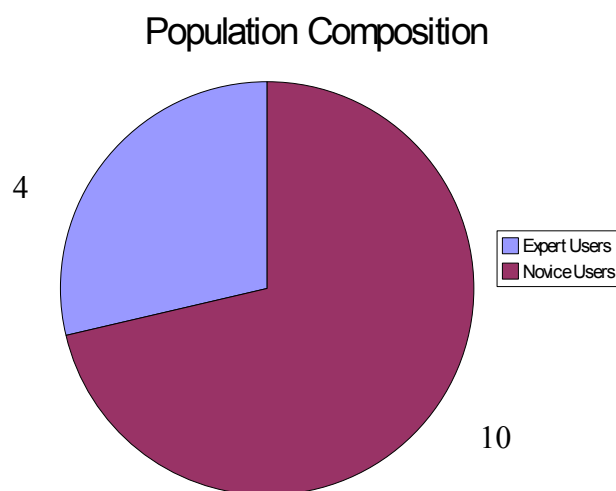
# Chapter 6

## 6 Data Analysis and Results of the Experiment

### 6.1 Overview

This chapter presents and analyses the results from the first set of user experiments of the MIRMAid interface, whose set-up was explained in the preceding chapter.

### 6.2 Sample population analysis



*Figure 6.1: Population Composition*

Figure 6.1 shows the population composition of all the test subjects in the user experiments. There were two groups of users that took part in the user experiments. The first group were expert users from the Music Technology Masters students at Stellenbosch University. The second group were Masters students in the Computer Science department at UCT.

There were four expert users who took part in the user experiment from Stellenbosch University.

All the users from the Stellenbosch University testing venue were assumed to be expert users. This assumption was later confirmed through the responses to the questions that asked if test subjects either used or wrote music manipulation software. This method allowed users to be classified as expert users as well. All 4 test subjects at the Stellenbosch University test venue were confirmed to be expert users. They all had formal music training and rated themselves as either good or fair on the computer proficiency question.

*Table 6.1 : Summary of user profile in terms of music and computer training.*

|                |          | Good | Fair |
|----------------|----------|------|------|
| Music Training | None     | 4    | 0    |
|                | Informal | 6    | 0    |
|                | Formal   | 3    | 1    |

Table 6.1 shows the composition of the sample population in terms of musical training and computer proficiency from both testing venues and includes all 14 test subjects who took part in the experiment. All the test subjects at the UCT test venue either fell into "None" or "Informal" music training categories, which means that most of the test subjects overall had elementary knowledge of music. There were no users at UCT who were re-classified as expert users. All the test subjects were proficient in using computers.

All the participants who participated in the user experiments were between the ages of 18 and 35. All the equipment was identical in both test venues and the test venues themselves were very similar. The test venues did not have any obvious impact on how the experiment was conducted or on the results that were returned from them.



## 6.3 Results Summary

Table 6.2: Results summary from the questionnaire

|   | Strongly agree | Agree | Neutral | Disagree | Strongly disagree |
|---|----------------|-------|---------|----------|-------------------|
| The task was difficult                        | 0              | 1     | 1       | 7        | 5                 |
| I understood instructions                     | 2              | 7     | 2       | 3        | 0                 |
| Group 1                                       |                |       |         |          |                   |
| I got stuck                                   | 0              | 7     | 2       | 2        | 3                 |
| I Accomplished both tasks                     | 0              | 9     | 4       | 1        | 0                 |
| I requested help                              | 0              | 8     | 3       | 2        | 1                 |
| Group 2                                       |                |       |         |          |                   |
| The interface was difficult to learn          | 0              | 1     | 3       | 8        | 2                 |
| The interface does it's job well              | 2              | 8     | 3       | 0        | 1                 |
| I know where I am in the interface            | 1              | 7     | 1       | 4        | 1                 |
| I know at which step I am in process          | 1              | 6     | 3       | 3        | 1                 |
| I know how to execute all steps required      | 3              | 6     | 5       | 0        | 0                 |
| I can correct mistakes                        | 1              | 5     | 3       | 5        | 0                 |
| I get lost in the interface                   | 0              | 2     | 3       | 6        | 3                 |
| The feedback is helpful                       | 1              | 5     | 5       | 2        | 1                 |
| Actions for transformation selection is clear | 1              | 10    | 1       | 2        | 0                 |
| Group 3                                       |                |       |         |          |                   |
|   | Excellent      | Good  | Average | Poor     | Very poor         |
| Intuitiveness                                 | 0              | 8     | 4       | 1        | 1                 |
| Overall                                       | 0              | 9     | 4       | 1        | 0                 |
| Layout  | 2              | 6     | 4       | 2        | 0                 |

Table 6.2 summarises all the responses from the questionnaire from all the test subjects who took part in the user experiments. In the first row of the table there is a non-numeric scale of five different alternative ratings ranging from "strongly agree" to "strongly disagree", corresponding to the rating system that was used in the questionnaire by test subjects to react to the statements they were presented with.

The feedback from the questionnaire can be grouped into three different categories. The first group present responses from the task-based questions. This group included questions that enquired if the instructions given to perform the experiment were understandable and if the tasks were perceived difficult. Responses from this category were important, as they indicated the weighting that should be placed on the effect of task difficulty on the execution of the task and the

performance of the interface.

Most of the participants indicated that they had no problems understanding the instructions to the task. There were one participant who indicated he had problems with understanding the task itself. There was one participant who found the tasks difficult - the rest of the participants (both novice and expert users) did not find the tasks difficult.

In contrast to this many test subjects indicated that they had to request help from the experimenter. All the test subjects indicated that they got stuck somewhere in the interface. In contrast to this most test subjects were able to complete both tasks successfully. Only one of the test subject indicated that he was unable to complete both tasks successfully.

The second group of responses correspond to interface and experiment based questions on the interface. The test subjects rated the interface well on all of the interface based questions.

Within this group the statement responses can be further subdivided into two groups. The first group are statements that generally measure performance of the interface on certain concrete aspects, and the second group of statements are more indirect or more abstract.

The third group of responses represents opinions on three major aspects of the interface - navigation, layout and intuitiveness. A different scale is used here, ranging from "excellent" to "very poor".

Some responses from questions are combined into one composite measurement for one design goal. Together they give a more reliable overview of the design goal than would have been possible otherwise by taking individual questions as goal proxies. This increases the reliability of the results obtained through the interface.

## **6.4 Discrepancies between results from task based and interface based statements**

There is an interesting relationship between the responses to most of the interface based statements in general and the task based statements. This is particularly true for the two statements: "I got stuck somewhere in the interface" and "I needed help". In all the interface based questions the test subjects rated the interface extremely favourably on most of the interface based

questions, yet most of the test subjects indicated that they got stuck in the interface somewhere and that they needed assistance with something in the interface.

We decided to investigate this matter by going back to the open ended question in the questionnaire that asked where subjects got stuck, and grouped the responses into different categories: responses given by expert users and responses given by novice users.

Amongst expert users, there were two test subjects who did not get stuck in the interface. There was one subject who had a neutral response. He gave the location of the error as being at the "Load Transformations" button when trying to execute a query in the second task. The other test subject got stuck in the interface because of confusion between numbering used on the interface and on the task instructions.

All the novice users got stuck somewhere in the interface except for two test subjects who disagreed with the statement. The test subjects who agreed with the statement could be divided into three groups: those who had problems with the naming between the interface and the task; those who had problems with the navigation while performing the second task; and an execution problem with the interface.

The confusion in numbering between the task instructions and the interface was an unexpected result of making the task description as broad as possible. The problem is that most subjects assumed that that steps to be taken for the tasks in the task description are steps as opposed to guidelines for figuring out how to execute the tasks themselves.

Most of the testers expected that the instructions would be exactly set out for them, indicating that there was not enough clarity in explaining this before they started the user experiments. Some of the novice users did not understand the context of the experiments properly - they just skimmed over the explanation of the project.

## **6.5 Evaluation of design results**

### ***6.5.1 Measurement of design goals***

The design goals against which this interface was measured were set out in the beginning of the interface creation process. These goals were: simplicity, usability and adequacy.

### 6.5.1.1 Adequacy

Adequacy is how sufficiently the tasks required for the user experiments were performed.

The adequacy of the interface was measured by the total number of people who completed both the tasks that they were set. There was a question on the questionnaire that asked whether or not the test subjects completed the tasks. Most of the test subjects indicated that they were able to complete both tasks as is illustrated in Figure 6.2.

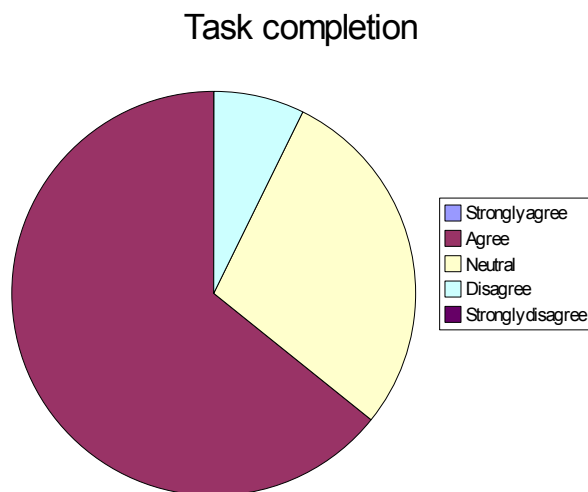


Figure 6.2: Indicates the percentage of subjects who agreed with the question that they were able to complete the task that they were set.

The second measure of adequacy was the responses to the question on whether the interface "does its job well".

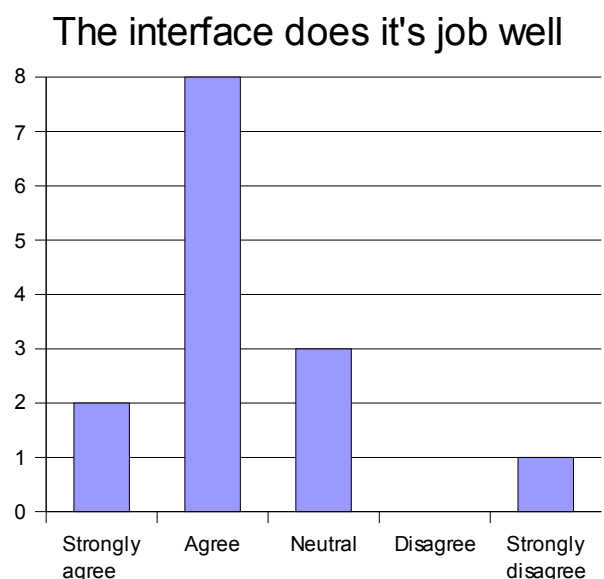


Figure 6.3: Results from the the interface does its job well statement.

On this question most of the test subjects agreed that the interface was able to do its job well as can be seen in figure 6.3.

### 6.5.1.2 Simplicity

Simplicity is how easy the interface is to use.

Simplicity was measured using learnability and intuitiveness. Learnability was measured by the responses to the direct question, "it was difficult to learn the interface". Intuitiveness was tested by the responses to three questions: 1) If the test subjects got stuck; 2) if the test subjects needed help, and 3) if test subjects knew what to do at every step of the query process.

#### 6.5.1.2.1 Intutiveness

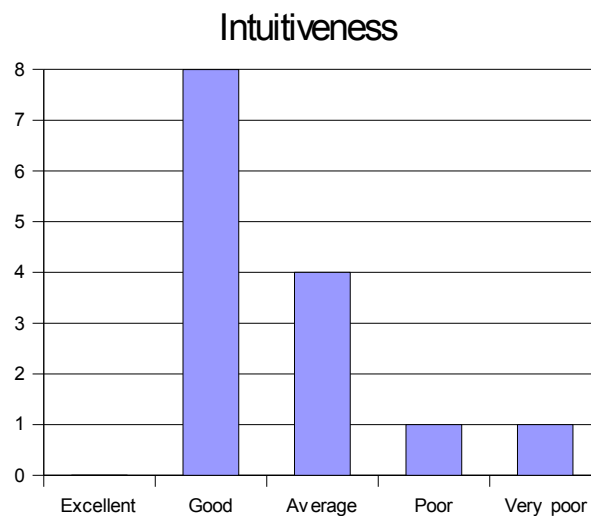


Figure 6.4 : This is a graph of the responses to the intuitiveness question in the questionnaire.

Figure 6.4 gives a summary of responses to the intuitiveness question in the questionnaire. Intuitiveness is the measure of how easily the interface can be used only by guidance given through the layout and structure of the interface without the help of any additional instructions. Users were asked to directly rate intuitiveness on a scale ranging form very poor to excellent. Most test subjects rated the intuitiveness of the interface on the scale as good or average.

Another important indication of intuitiveness was if the test subjects were sure what do at every step of the query process. "I know how to execute all tasks required" was the statement on the questionnaire that measured this. All of the responses to this question were either in the positive or neutral as can be seen in table 6.2.

#### 6.5.1.2.2 Learnability

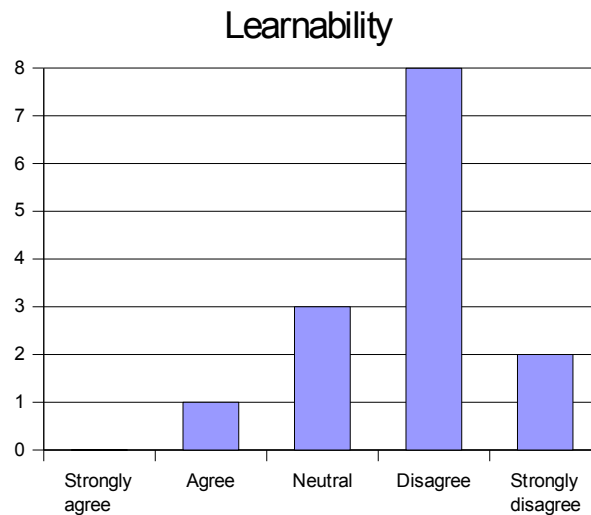


Figure 6.5: Graph of responses from the question if the interface was difficult to learn or not.

There were many subjects who agreed with the statement that the interface was easy to learn, as is illustrated in Figure 6.5.

The other test subjects who gave structure a poor or average rating qualified their decision by saying that there were problems with common conventions that they would have expected from a works pace.

#### 6.5.1.3 Usability

Usability is a measurement for how effectively users are able to use the interface. Usability of the interface was measured by navigation, layout and structure.

Navigation was tested by using various navigation questions asked in the questionnaire, and a direct question on how subjects viewed the navigation of the interface. Subjects were also asked their opinion on layout and structure of the interface directly. Other usability criteria include questions about if the task could be adequately be accomplished or not and if subjects were able

to correct mistakes they made while using the interface. There was a general question on usability as well.

Good layout also has a big effect on usability of the interface.

### 6.5.1.3.1 Layout

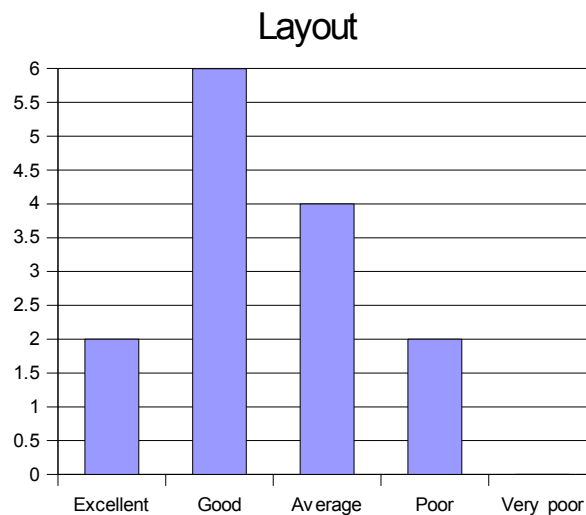
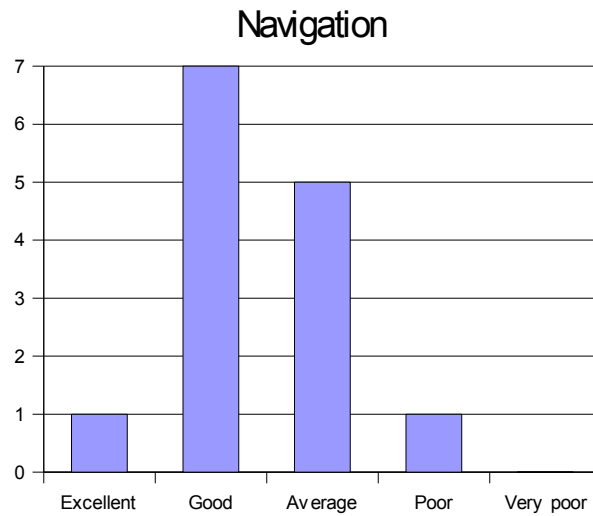


Figure 6.6 : This is a graph to the responses from the layout question in the questionnaire.

Figure 6.6 gives a summary of responses to the layout question in the questionnaire. Layout is closely related to structure and together has a cumulative effect on navigation and general intuitiveness of the interface.

Most test subjects' responses to the layout of the interface was either good or average with only two test subjects giving the layout of the interface poor ratings. Another good indication of the effectiveness of layout are the responses to the statement, "I know in which step I am in the process". Layout should indicate where users are in the interface by just how the elements are arranged. There were seven positive responses to the question, three neutral responses and four negative responses to the question. Although there were many comments about layout in the comments section, the layout was overall judged as mostly "average" or "good".

### 6.5.1.3.2 Navigation



*Figure 6. 7 : This is a graph of the responses to the navigation question in the questionnaire.*

Figure 6.7 gives a summary of responses to the navigation question in the questionnaire. Navigation was tested by using various navigation related questions in the questionnaire. The main measure though was asking testers to rate the navigation of the interface directly.

The "I get lost in the interface" question indicates that somewhere in the interface there is a breakdown in navigation for some of the test subjects. "I know where I am in the interface" and "I know which step I am at in the interface" questions measures transparency of the navigation. A question that measured the flexibility in navigating through the interface was if test subjects had the ability to correct a parameter that was set previously in the interface.

On all of navigation based questions the responses were mostly neutral. An unexpected confounding factor in the navigation was that some of the testers assumed that the numbering on the task sheet corresponded directly with the numbers on the interface. This caused some of the testers to look for items in the interface prematurely or skip steps that were necessary for them to complete the task that they were set.

This specific issue contributed to the number of people who needed to ask for help while performing the user experiments. This information was derived from the open-ended questions at the end of the questionnaire and will be discussed later in this chapter.



## 6.6 Problems with the MIRMaId interface

Problems with the interface can be grouped based on feedback from the users during the user testing and comments from the questionnaires.

### 6.6.1 Navigation

Navigation was a weakness that can be improved on in future versions of the interface. The primary problem was a complicated navigation path across the interface. Figure 6.8 below shows the easiest navigation path to follow to perform the second task set for the experiments. The green arrows indicate the direction of the steps that were executed and the numbers show the sequence in which actions on the interface were performed.

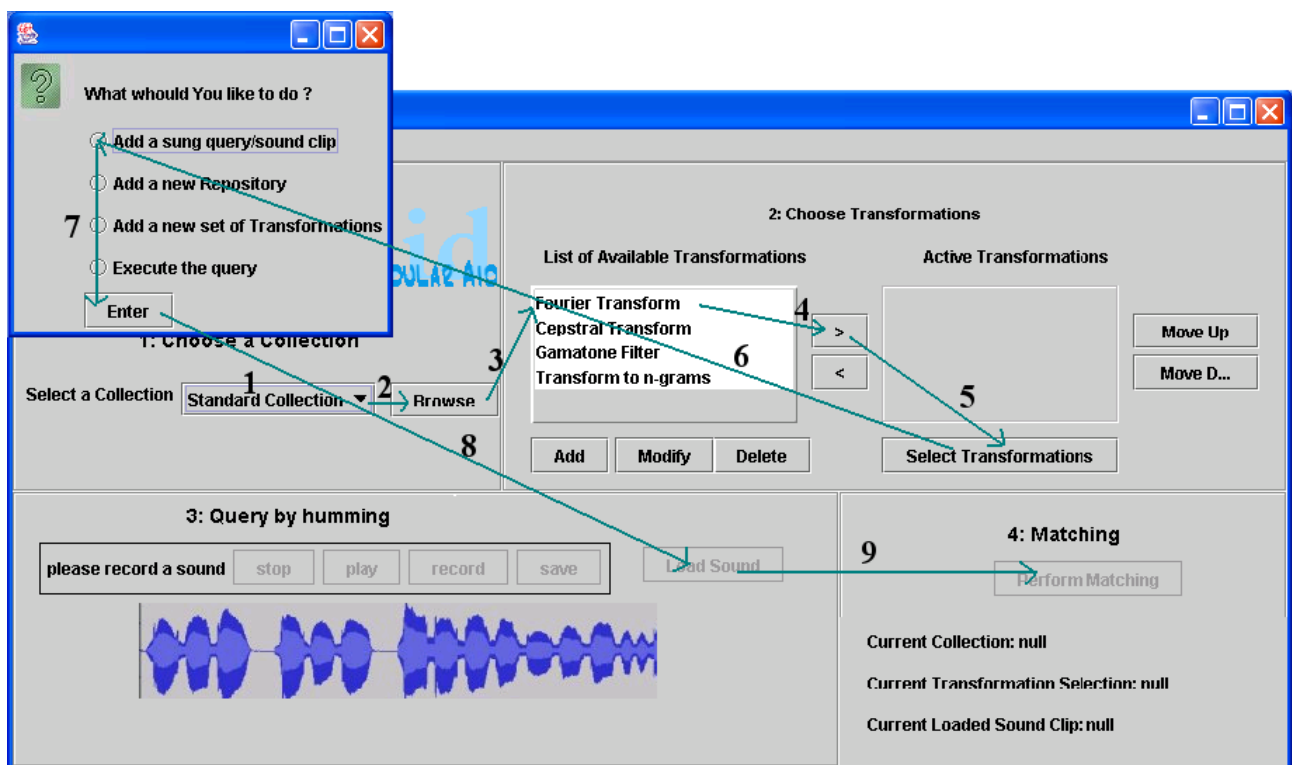


Figure 6.8 : This figure illustrates the current navigation path of the interface.

Although jumping in the navigation was minimised drastically from the first iteration of the interface, two jumps remained and caused many of the test subjects to request help as can be seen in Table 6.2. The navigation did not flow from one frame to the other linearly (illustrated in Figure 6.8).

Average user navigation patterns were tracked by observing their actions when performing the tasks. They began navigation at the correct element at the top left hand corner of the interface,

then had problems moving on to the next element. Test subjects pressed random buttons when they were confused over whether or not they have already completed a step or not, The same happened when they were unable to execute a step that they believed they would be at in the interface. This prevented them from executing steps in sequence.

Firstly, all the test subjects seemed to skip past the browse button in the first frame and only return to the button after they have re-read the instructions or tried to access the transformations frame unsuccessfully.

The next navigational weakness was a big jump after pressing the "Select Transformation" button. The pop-up screen launched in the top right corner of the screen which forced the test subjects to backtrack to previous elements in the interface. Initially this element was added to provide users an option for adding additional repositories of queries. This disorientation was compounded when the selection was made on the pop-up screen and the pop-up screen closed. There were then no cues on where to start the next step in the process. A contributing factor in this confusion is that the terminology and naming differed in the interface and the task instructions. When test subjects found their spot in the interface again, they were generally able to continue and complete the task without any further intervention from me or searching on the task paper.

People wanted the option of returning back to the main interface after one of the results frames were shown.

### **6.6.2 General layout and operation logic**

A disproportionate number of testers who got lost in the interface at some point ended up searching for different buttons and elements on the interface. Although there was error trapping in place for pressing the wrong buttons at random, it was very frustrating for the test subjects.

At start up all the elements on the interface have equal focus and weighting. The layout is asymmetrical as well. Therefore layout may appear disorganised and overwhelming to some users because they are overloaded with information on the interface.

There was a duplicate step when executing a query for the second experiment that was not picked up or corrected during the pilot test. This affected the overall navigation of the interface negatively. The execute query option appeared twice in the process of the second user experiment. The user

had to press it on the pop-up form and on the main screen to execute the query.

There was no Back or Cancel button allowing test subjects to cancel the current query process on the interface after it has been started, or to modify any data before the query is executed.

There was also no consistency in moving from one frame to other frames. This contributed to test subjects having problems moving from one frame element to other frame elements in a set order, especially when moving from the first frame element to the second frame. Test subjects did not realise that they had to press "ENTER", before they could move to the next frame. This was a limitation of the prototype and a production interface should be able to navigate controls more intuitively. Some test subjects even went to the menu to search for a way to move on to the next frame. The presence of the "ENTER" button was a limitation of the prototype and a production interface should be able to activate controls in a more intuitive and consistent manner.

### ***6.6.3 Controlling query options***

Controlling query options is the most consistent problem uncovered during the experimentation process by the interaction of the test subjects with the interface. Query options are controlled through a pop-up screen that appears at the top right hand corner of the interface.

A contributing factor was that this pop-up screen was not directly mentioned in the instructions to the tasks. When the test subjects encountered the pop-up screen the purpose of the screen was not immediately obvious, and test subjects then went back to the experiment instructions. After this they asked for help. In the second task a lot less people had this problem again, indicating that the problem was due to the instructions and not the interface.

The physical layout of the interface presented problems because all the elements had the same visual weighting, confusing test subjects to where to start the tasks necessary for the query. The logical layout of the interface created many problems as well, although all the areas in the interface were numbered.

### ***6.6.4 Finding and Loading Sound Clips***

One consistent observation from all the users is that they were all searching all over the interface for the button to load sound clips. Again, the elements was named differently on the interface than on the instructions.

## 6.7 Summary

From the experimentation process it can be concluded that 1) The users felt quite comfortable using the interface 2) The interface was useful and usable and 3) There is a lot of scope for generalising and extending the work.

The were positive results returned from the questionnaire in terms of both task-based questions and interface-based questions, indicating that in general the interface succeeded in the goals that were set out for it at the beginning of the design process.

The user experimentation process also revealed a few gaps in the navigation and the program flow of the interface. The cause of these problems were investigated and found to be a combination between a mismatch in the numbering of the interface and the instruction sheet and the interruption of workflow presented by the pop-up window.

The results from the experiments also highlighted the need for further experimentation to refine the interface.

# Chapter 7

## 7 Future Work

### 7.1 Overview

This chapter presents some future refinements to the interface and gives some details on one possible implementation of the test-bed.

This chapter also gives possible enhancements to the interface that would be needed to convert the interface into a conversion/portal tool that can assist frameworks to access classes and methods from other smaller frameworks easily, cleanly and transparently.

### 7.2 Interface Enhancements

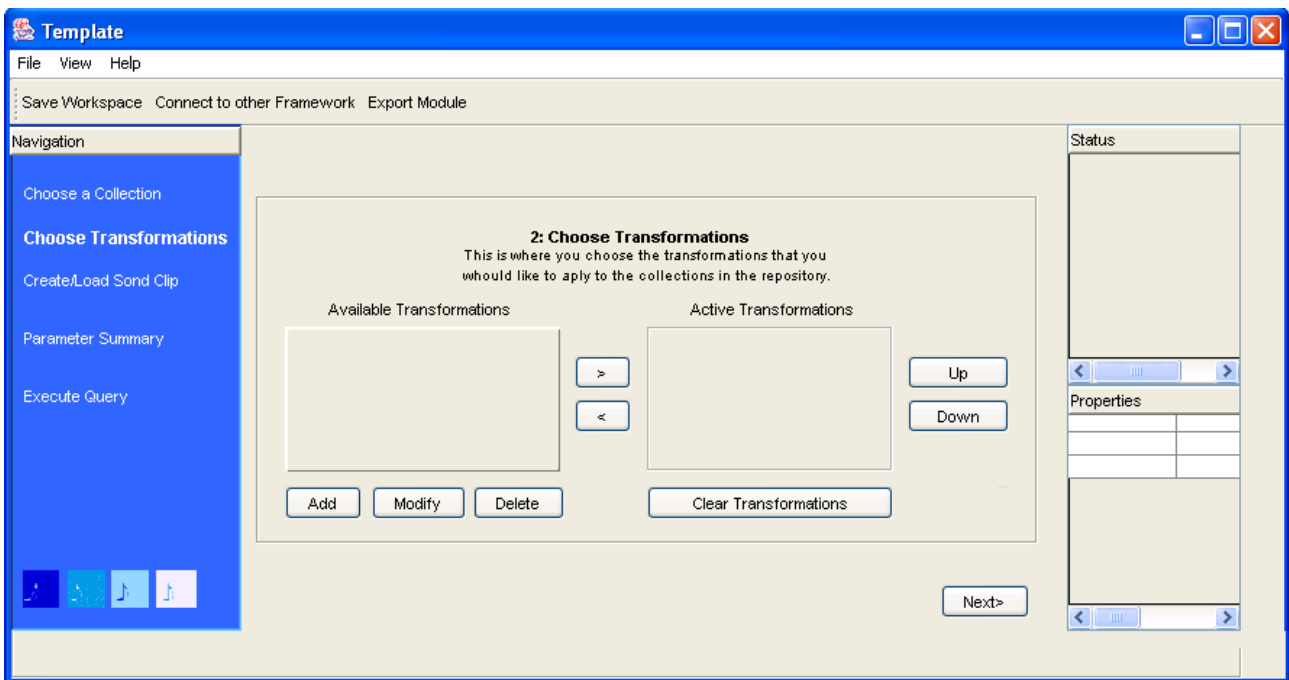


Figure 7. 1: Image of how the interface could look like in the future.

The interface (Figure 7.1) is a skeleton prototype of what the interface might look like in the future. The prototype reflects the suggestions and observations that were made during user testing.

The navigation elements were separated from the form functionality and were replicated uniformly

throughout the rest of the frames.

There was an additional tool bar added to the top of the interface. On the toolbar panel there is still a lot of space to add additional functions.

The interface still kept the component windows but elements were shifted around and only one element is shown at a time to reduce visual clutter.

This iteration of the interface also attempted to conform to the standard layout formats of well known workspaces.

Common elements from other frameworks, like command line utilities, may be included to minimise the learning curve for users.

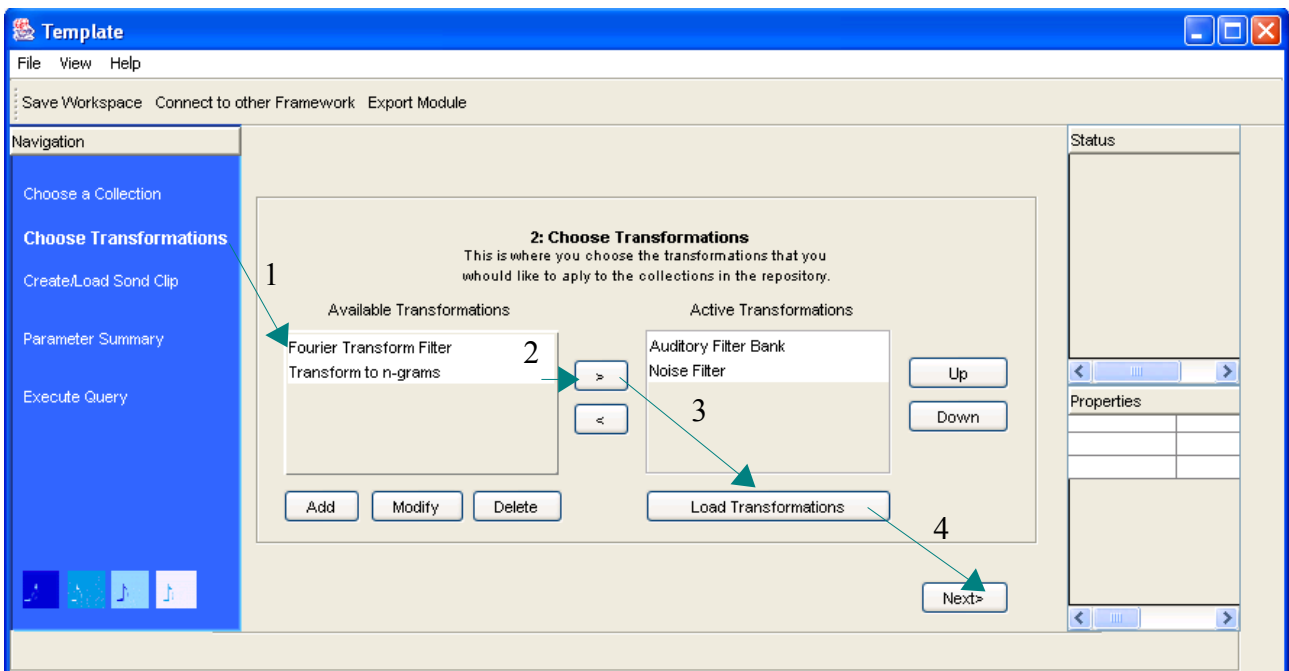


Figure 7.2: New navigational path over one frame

The greatest challenge to refining the interface after feedback from the first round of experimentation was providing guided navigation. Guided navigation will have a big impact on the quality and simplicity of the interface in the future.

Subtle changes in font and colour variations can be used to differentiate active areas in the workspace to provide visual guidance to the user.

More information was also requested by many of the experimenters. This could prove important because this would communicate to users where they are in the process and their location in relation to broader systems. In response, better feedback should be added by providing a window in which users can see what tasks they have already completed.

A “wizard” type of interface could also be used to break the task down to a number of smaller interfaces shown to the user in a sequence. This could have been another way to solve navigational problems. This was not considered for the initial interface, since observations were needed on how people engaged with the interface, as opposed to automatically clicking the “next” button at each screen of a “wizard” type interface.

The interface may be improved by minimising the dependence of the test-bed on importing externally created modules by adding functionality for modules to be written in MATLAB within the interface. This could be done by adding a MATLAB command line utility that will allow MATLAB modules to be executed from the command line by MATLAB. This will allow users to create and run MATLAB modules within the test-bed.

Another improvement to the functionality of the workspace would be to add a mechanism that could allow the works pace to access data from specialised repositories and collections from different sources on-line. An important aspect of this would be to provide authentication mechanism to stop unauthorised access to data that is under copy protection.

This will allow the interface to fit into a broader architecture for a large scale MIR/MDL testing and development environment as outlined in the MIR/MOL evaluation project white paper collection [Downie, 2003].

On the results frame that is returned from the query there will be an area where there will be support for browsing.

It would also be beneficial if the interface could allow users to switch between different modes of evaluation or be able to convert a query made in one mode into another mode just by supplying additional information.

The framework could support exporting successful combinations of modules that were evaluated in the test-bed. Packaging the modules could make it easier for modules to be reintroduced back into their originating framework or imported into compatible frameworks.

The interface can be changed so that it could indirectly link to each other through the workspace.

It is suggested that the test-bed should avoid providing fully functional classes or replicating classes found in other frameworks.

The interface could produce specifications for systems from the output that is returned for various tests.

### **7.3 Data Collections**

There is an undue emphasis on Western Music Audio representations in Music Information Retrieval. Audio features are assumed to be culturally neutral but so far there has been very few attempts to test this hypothesis [Futrelle & Downie, 2003]. The first change to the interface may be to add a music repository. The repository could contain a number of collections - ranging from western standard music to collections that contain specialist collections of Southern African music that can be obtained from different sources, like the Contemporary Arts and Music Archive [CAMA, no date].

CAMA contains both video and audio data as live recordings and studio recordings. At the moment the records are only available from the African Studies library at UCT. The scope of the Southern African collection may be broadened to include other forms of Southern African Music like Kwaito if there are not enough sound clips available.

The one big disadvantage is that the database cannot be distributed since most of the music is under copyright.

The interface could easily be extended to accept different types of data besides audio content. The simplest departure would be to add symbolic data so that score reading/"optical recognition" algorithm testing can also be introduced. Audio files that have other data, like bibliographic data, attached can also be added.

The modified interface would also be able to handle testing modules for MIDI, video and text data. Having different kinds of data in the test-bed could also allow different content based and text based information retrieval methods to be tested against each other. In the future the interface could support multi-modal testing. Different types of data that are available from one sound clip can be tested. Tests can also be done for exploiting the relationship between different representations



of the same piece. [Downie,2003] and for more effective multi-modal retrieval.

To allow modules to act uniformly on all clips when they are converted into different annotations, audio clips in music collections are usually converted into one common file format, e.g. .wav file format at 44.1 KHz, 16 bits per second clips [Downie,2004].

The problem is that after clips are converted, they need to be pre-processed, to eliminate the noise from the recording that might have inadvertently been included because of conversion to electronic form or transfer from one format to another.

A repercussion of pre-processing is that the results returned from the pre-processed collection may become skewed, because of error that will be introduced when re-sampling and converting the clips from their original format.

The framework could be improved by dealing more gracefully with music clips of different sampling rates, reflecting the differently formatted data present in the different repositories, similar to how it would be in the real world.

## **7.4 Future testing of the test-bed and the interface**

In order for testing of the interface to be most meaningful, various evaluation tasks from the Music Information Retrieval Evaluation Exchange(MIREX) 2005 competition could be duplicated for testing the interface and the test-bed. These tasks can include audio melody extraction, audio music similarity and retrieval and audio tempo extraction [MIREX, 2005].

Modules used for these tasks may also be used and should include modules like those for audio engineering and digital signal processing.

Once test modules have been imported another stage of user experimentation could follow. In this round of user experimentation all the features indicated on the interface would be fully functional. This round can then include more thorough measurements on timing, incidence of error and the frequency of different task-based errors like failed actions.

The sample population could be modified to include a larger segment of the MIR community. This may be done by setting up a website which can contain the interface as a Web service and allow

users to interact with the test-bed. An alternative would be to test the interface through a Web questionnaire. The second way would be to modify the interface so that it is plug-in to M2K, and users are requested to fill in a Web questionnaire.

Support could be added to the interface for creating comparative evaluations for assessing multiple algorithms, for one retrieval task.

Support could also be added to allow users to do related queries and compare them with ones they have already done or compare results from different evaluation modes.

In choosing external evaluation techniques there is subjective bias introduced and can be over fitted [Futrelle,2003]. In addition evaluation results change in response to using different assessments [Voorhees,2004]. This could be improved by adding test collections in addition to data collections in the interface. This will also allow different metrics to be used for evaluating different retrieval strategies for one specific information retrieval task.

This will allow the evaluation of different combinations of transformations and retrieval strategies but each time you use a different way to test the effectiveness after which you can set up a matrix to compare different retrieval strategies against each other.

This could be used to set up an evolutionary tournament for different test metrics on specific retrieval tasks. The rules of the game can change and different results will be yielded. Different combinations could be tested on multiple criteria and then the different combinations and variations are tested through an evolutionary game.

## **7.5 Summary**

Since most of the dissertation discussed the interface for a conceptual test bed, this chapter concentrated more on contextualising implementations of a test-bed as well as possible extensions to the interface and exploring paths that were not pursued in this dissertation.

This chapter further investigated ways in which the interface could be extended to support functionality that would make the interface useful even if it is incorporated into larger frameworks as can be done with a proposed international music retrieval test-bed [Downie,2003].

# Chapter 8

## ***8. Conclusion***

This dissertation presented an interface for a Music Information Retrieval (MIR) test-bed in order to investigate the composition of algorithms for music manipulation. The interface allowed users to combine modules from different frameworks by comparing different sequences of modules to find optimal combinations for specific problem domains.

The interface was built using an iterative process consisting of a combined analysis and consultative stage with users, after which the interface was modified and refined. We approached the interface building from the perspective of a total novice user, to make sure that it would be as simple as possible.

The interface was then subjected to a set of user experiments. The user experiments were designed to test the interface using appropriate tasks. Test subjects were then asked to complete a questionnaire which required them to rate some interface and task based statements. The interface was tested on its compliance with three design goals that were set at the beginning of the design process. These design goals were adequacy, usability and simplicity.

We determined that on adequacy, both expert and novice users rated the interface well. On simplicity, the interface was also rated as good. Although there were a few problems associated with navigation and layout, the interface still performed reasonably well in terms of usability.

Most of the results on the different design goals were gained from the interface based questions. From these responses it can be concluded that the interface performed reasonably well with expert users and novice users on interface based statements.

Task based statements tested operational and concrete aspects of the interface. On task based statements there were many interesting results returned from the questionnaire. The most interesting results were from the statement which asked if test subjects needed to request help from the experimenter. While the usability of the interface was rated by the test subjects as good, all the test subjects had to request help with the interface.

This anomaly between the task based statements and the interface based statements was

investigated. It was revealed that the discrepancy was partially due to an extra step that was omitted in the task instructions, and partially due to a mismatch between the numbering on the task instructions and the numbering on the interface. It was also seen that interruptions to the workflow of the interface was problematic and should be avoided.

A prototype skeleton of how the interface can look in the future was also created. It was modified to reflect the results and comments gained from the user experiments. Furthermore, suggestions were also given on how the interface can be expanded and generalised. This included suggestions on how the interface can be enhanced and extended in different ways. This included suggestions on extending the functionality of the interface to include tools for: 1) writing modules within the interface; 2) accessing classes and methods from other frameworks, and 3) accessing data from specialised collections from different sources on-line.

The scope of the interface can be broadened by accepting different types of data besides audio content. Having different kinds of data in the test-bed could also allow different content based and text based information retrieval methods to be tested against each other and open the possibility for multi-modal testing.

Another key enhancement that was suggested was to allow users to switch between different modes of evaluation. The implications are that: 1) Support could be added for creating comparative evaluations for assessing multiple algorithms, for one retrieval task; 2) Support could be added to allow users to do related queries and compare them with ones they have already done or compare results from different evaluation modes, and 3) Support could be added for creating comparative metrics for comparing the effectiveness of different metrics in evaluating different retrieval strategies for one specific information retrieval task.

Furthermore the framework could support exporting successful combinations of modules that were evaluated in the test-bed and produce specifications for systems from the output that is returned for various tests.

# Appendices

## ***A1 Task explanation used during the project***

### About the experiment

The user experiments were designed to test the relevance of the interface I have built, to see how easily transformations can be applied through the interface and to see if the interface is working properly. The other use of the experiments are to see how you interact with the interface by performing simple functional tasks.

The tasks that you are required to perform are quite simple tasks. The tasks will require you to perform set queries using the interface.

I am working on the design for an interface, and as part of the process I am asking a variety of people to attempt two tasks using it and to fill out a feedback questionnaire afterwards, to see what elements of the design need to be changed and to see if the interface is working properly.

In the first task involves querying the test-bed by using a sound clip from a well known composer for finding the list of records in the test-bed that satisfies the parameters that you had selected beforehand.

The second task involves selecting a set of parameters for extracting information from two different repositories to evaluate the difference in search performance between the two repositories.

After completing the tasks, you will then be required to fill out a questionnaire in which you will document your experiences with the interface and with the tasks you have just completed.

Please remember that you can withdraw from the experiment at any time.

Please Turn over the page for the first task...

## ***A2 Instructions to the first task used in the interface***

### **Task 1:**

The task for you is to perform a spot query on the test-bed by using a sound clip from the repository, in this case a clip from Vivaldi's Four Seasons. spot queries are done to see how an individual query performs when it is used with a set of transformation parameters. For this experiment you are asked to select two transformation parameters and a collection of music you would want to use for the query and then execute it.

How to perform the task:

#### **1. Select a collection**

In this step you will specify a collection to which you will apply transformations too. Select the "standard collection".

#### **2. Select and Load Transformations**

Select the "Auditory filter bank" and "noise filter" transformations. These are the transformations that will be applied to the collection you chose. Please make sure that the Auditory filter bank transformation is applied before the noise filter transformation.

#### **4. Load a Sound Clip**

Load a sample clip called 1.wav. The sample path is  
myMusic>ExperimentExample\_wavs>Classical> 1.wav

#### **5. Press the query button**

After this you should be presented with a screen in which lists possible matches to the query you have entered into the system. You can click on the table and listen to the returned clips to verify the validity of the results.

Exit the program when you are ready to do so.

That is it for the first task.

Please continue with the second task on the next page ...

## ***A3 Instructions to the second task used in the project***

### **Task 2:**

This task will require you to evaluate the search performance of a set of transformations on a single repository, this will show the performance of the transformations you chose did in matching.

How to perform the task:

1. Open the framework workspace

Double Click the program icon on the desktop that says "MIRMaid.jar"

2. Select the collections

Select the collections called standard collection. This is the collections to which you are going to apply transformations too.

3. Select transformations

Select the Auditory filter bank and Fourier filter transformations. These are the transformations that will be applied to the collection you chose. Please make sure that the Auditory filter bank transformation is applied before the Fourier filter transformation.

4. Perform the Query

Execute the query by pressing the "Perform Matching" button.

After this you should be presented with a screen which will show a graph which represents the matching efficiency of the current combination of transformation variables you have chosen.

Exit the program when you are ready to do so.

Thank you. That completes the tasks.

Please complete the questionnaire now, on the next Page...

### ***A3 Questionnaire used in the project*** **Questionnaire**

Please circle the response in each of the questions that is most applicable to you :

Subject Profile

Are you male or female?

- Female
- Male

Age(Please Select one):

- Under 19
- Between 18 – 36
- Over 36

What is the highest level of musical education you have

- None
- Informal
- Formal

Please rate your proficiency in using computers

- Good
- Fair
- Poor

Have you ever made use of music manipulation software?

- Yes
- No

Do you write music manipulation software?

- Yes
- No

Please indicate the extent to which you agree or disagree with the following statements

|  | Strongly<br>Agree        | Agree                    | Neutral                  | Dis-<br>agree            | Strongly<br>Disagree     |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| <u>Task Related Statements</u>             |                          |                          |                          |                          |                          |
| The task was difficult.                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I understood the instructions to the task. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| The interface was difficult to learn.      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I got stuck somewhere in the interface.    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Please indicate where you got stuck.       | .....                    |                          |                          |                          |                          |
| I accomplish the task set easily.          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I needed help within the interface.        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |



|   | Strongly Agree           | Agree                    | Neutral                  | Dis-agree                | Strongly Disagree        |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| <u>Interface related Statements</u>                     |                          |                          |                          |                          |                          |
| The interface does its job well.                        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I know at all times where I am in the interface.        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I know at which step of the process.                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I knew how to execute all the steps required            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| It was easy to correct mistakes made in other steps.    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| I get lost in the interface                             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Feedback and suggestions from the interface is helpful  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Actions necessary for transformation selection is clear | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

|  | Excellent                | Good                     | Average                  | Poor                     | Very poor                |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| <u>Interface Related Questions</u>                 |                          |                          |                          |                          |                          |
| How do you rate the intuitiveness of the interface | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| How did you find the interface overall             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| How do you rate the layout of the interface        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| How do you rate the navigation of the interface    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

If there is anything that you want to change in the interface what would that be ?

---



---



---



---

Do you have any comments ?

---



---



---

Thank You for your Time,  
Candice

## Bibliography

- Amatriain,X. Arumi,P. (2005). Developing Cross-Platform audio and music applications with the CLAM framework. Proceedings of International Computer Music Conference 2005. [online] Available <http://www.iaa.upf.edu/mtg/publications/9d0455-icmc05-clam.pdf> [2006, 16 February].
- Amatriain,X. (2004). An Object-Orientated metamodel for Digital Signal Processing with a focus on Audio and Music Ph.D. Dissertation. UPF. Barcelona
- Bainbridge,D. Cunningham, S. Downie, J. (2004). Greenstone as a music digital library toolkit. In Int. Symposium on Music Retrieval (ISMIR) 2004 Proceedings.42–43.
- Battle, E. and P. Cano (2000). Automatic Segmentation for Music Classification using Competitive Hidden Markov Models. In Int. Symposium on Music Retrieval (ISMIR)2000. [online]. Available [www.iaa.upf.es/mtg/publications/ismir2000-eloi.pdf](http://www.iaa.upf.es/mtg/publications/ismir2000-eloi.pdf) [2006, 16 February].
- Binu ,M. Davis, A. Zhen, F. (2003). A low-power accelerator for the SPHINX 3 speech recognition system, Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems, October 30-November 01, 2003, San Jose, California, USA
- Birmigam,W.P. Dannenberg,R.B. Wakefield,G.H. Bartsch,M. Bykowski,D. Mazzoni,D. Meek,K. Mellody,M. Rand,W. (2003). MUSART: Music Retrieval Via Aural Queries. In Int. Symposium on Music Retrieval (ISMIR) 2001. [online]. Available [ismir2001.indiana.edu/pdf/birmingham.pdf](http://ismir2001.indiana.edu/pdf/birmingham.pdf) [2006, 16 February].
- Boulanger,R.(2005).CSound. [online]. Available <http://www.csounds.com/whatis/index.html> [2006. 16 February].
- Byrd, D. Crawford, T. (2002). Problems of Music Information Retrieval in the Real World. Information Processing and Management 38:249-272.
- Cano,P. Battle, E. Gomez, E. de CT Gomes, L. Bonnet, M. (2005). Audio Fingerprinting: Concepts And Applications, Studies in Computational Intelligence (SCI) 2, Springer-Verlag Berlin Heidelberg.233-245.
- CAMA.(no date). Contemporary Arts and Music Archive. [online]. Available <http://cama.org.za> [2006. 16 February].
- Crawford,T.Iliopoulos,C.S. Raman,R. (1998). String matching techniques for musical similarity and melodic recognition. Computing in Musicology, Vol. 11: 73-100.
- Dannenberg,R. Hu,N. (2004).Understanding search performance in query-by-humming systems. In Int. Symposium on Music Retrieval (ISMIR)2004.[online]. Available [www.isimr.org](http://www.isimr.org) [2006, 16 February].
- Dannenberg, R, Birmingham, W, Tzanetakis, G. Meek, C. Hu, N. Pardo, B.(2004).The MUSART Testbed for Query-by-Humming Evaluation Computer Music Journal archive Vol 28(2) : 34 - 48
- Downie,J.S. (2003). Music information retrieval ,Chapter 7. In Annual Review of Information Science and Technology 37, ed. Blaise Cronin, Medford, NJ: Information Today.295-340.
- Downie,J.S. (2003).The TREC-Like Evaluation of Music IR Systems, The MIR/MDL Evaluation Project White Paper Collection, Establishing Music Information Retrieval (MIR) and Music Digital Library (MOL) Evaluation Frameworks: Preliminary Foundations and Infrastructures,12-16.
- Downie,J.S. Futrelle,J.Tcheng,D. (2003).The International Music Information Retrieval Systems Evaluation Laboratory:Governance, Access and Security. The MIR/MDL Evaluation Project White Paper Collection, Establishing Music Information Retrieval (MIR) and Music Digital Library (MDL) Evaluation Frameworks: Preliminary Foundations and Infrastructures,3-6.
- Downie,J.S. (2003).The TREC-Like Evaluation of Music IR Systems, The MIR/MOL Evaluation Project White

Paper Collection, Establishing Music Information Retrieval (MIR) and Music Digital Library (MDL) Evaluation Frameworks: Preliminary Foundations and Infrastructures Downie, J.S. [editor]

- UPF. (no date). CLAM User and Development documentation, Release 0.7.0, Revision 3. [online]. Available <http://www.iaa.upf.es/mtg/clam> [2006, 16 February].
- Eaton, J. (1998). Octave FAQ : Frequently asked questions about Octave. [online]. Available <http://www.octave.org/FAQ.html> [2006, 16 February].
- Flexer, A. (2005). Statistical Evaluation of Music Information Retrieval Experiments, Technical Report, Oesterreichisches Forschungsinstitut fuer Artificial Intelligence, Wien, TR-2005-18. [online]. Available <http://www.ofaLaUcgi-bin/tr-online00number+2005-18> [2006, 16 February].
- Futrelle, J. Downie, J.S. (2003). Interdisciplinary Research Issues in Music Information Retrieval: ISIMR 2000-2002, Journal of New Music Research, Vol 32, 121-131.
- Giorgi, Zoia, G. Zhou, R. Mattavelli, M. (2002). MPEG Audio Coding and XML: samples, models, descriptors. [online]. Available [www.lim.dico.uniml.it/umaxprojec/umax2002/docs/GZoiaMAX2002.Pdf](http://www.lim.dico.uniml.it/umaxprojec/umax2002/docs/GZoiaMAX2002.Pdf) [2006, 16 February]
- Gomez, E. Klapuri, A. Meudic, B. (2003). Melody Description and Extraction in the Context of Music Content Processing. Journal of New Music Research, 32(1).
- Gomez, E. Gouyon, F. Herrera, P. Amatriain, X. (2003). MPEG-7 for Content-based Music Processing. Proceedings of 4th WIAMIS-Special session on Audio Segmentation and Digital Music
- Gouvea, E.B. Chan, A. Mosur, R. (no date). Sphinx-3 s3.X Decoder (X=5). [online]. Available [http://cmusphinx.sourceforge.net/sphinx3/#sec\\_decoverview](http://cmusphinx.sourceforge.net/sphinx3/#sec_decoverview) [2006, 16 February].
- Griffin, T. (2001). Selected Writings. [online]. Available <http://tim.griffins.ca/writings/> [2006, 16 February].
- Haus, G., & Pollastri, E. (2001). An audio front end for query-by-humming systems. In Int. Symposium on Music Information Retrieval (ISMIR) 2001. 65-72.
- IMERSEL. (2004). International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL): Introducing D2K and M2K. [online]. Available [http://www.musicir.org/evaluation/m2k1v4\\_ISMIR2004\\_Handout.pdf](http://www.musicir.org/evaluation/m2k1v4_ISMIR2004_Handout.pdf) [2006, 16 February].
- IMIRSEL. (2004). The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) Project, [online]. Available <http://www.music-ir.org/evaluation/> [2006, 16 February].
- Karjalainen, M. Tolonen, T. (1999). Multi-pitch and periodicity analysis model for sound separation and auditory scene analysis. In *icassp*, Vol 2. 929-932.
- Klapuri, A. 2004. Signal processing methods for the automatic transcription of music. Doctoral Dissertation. Tampere, Finland: Tampere University of Technology. [online]. Available <http://sp.cs.tut.fi/publications/theses/doctoral/Klapuri2004.pdf> [2006, 16 February].
- Lamere, P. (2005). Tools we use, Version 1.5. [online]. Available <http://www.music-ir.org/evaluation/tools.html> [2006, 16 February].
- Mathworks. (no date). Matlab 7.0.4 Product Description. [online]. Available <http://www.mathworks.com/products/matlab/description1.html> [2006, 16 February].
- Mazzoni, D. Dannenberg, R. (2003). Melody Matching directly from Audio. Int. Symposium on Music Retrieval (ISMIR) 2003. [online]. Available [www.isimr.org](http://www.isimr.org) [2006, 16 February].
- National Instruments Corporation. (2006). What is LabWindows/CVI? [online]. Available <http://volt.ni.com/niwc/cvi/advanced.isp?node=11104> [2006, 16 February].

- Pardo, B. Birmingham, W. (2003). Query by Humming: How Good Can It Get?. The MIR/MDL Evaluation Project White Paper Collection, Establishing Music Information Retrieval (MIR) and Music Digital Library (MOL) Evaluation Frameworks: Preliminary Foundations and Infrastructures, 12-16.
- Puckette, M. (no date). Pure Data. [online]. Available <http://www.pure-data.org/> [2006, 16 February].
- Ravishankar, M. (2006). Sphinx-3 Guide. Available [http://cmusphinx.sourceforge.net/sphinx3/s3\\_overview.html](http://cmusphinx.sourceforge.net/sphinx3/s3_overview.html) [2006, 29 August].
- Reiss J. Sandler M. (2004), Audio Issues in MIR Evaluation. In Second International Symposium on Music Information Retrieval (ISMIR 2004) . [online]. Available <http://ismir2004.ismir.net/proceedings/p005-page-28-paper133.pdf> [2006, 29 August].
- Sun Microsystems Inc. (2000). JavaSound API Programmer's Guide, Chapter 2: Overview of the Sampled Package. [online]. Available <http://java.sun.com/j2se/1.4.2/docs/guide/sound/programmerguide/contents.html> [2006, 16 February].
- Selfridge-Field, E. (1998). Conceptual and Representational Issues in Melodic Comparison, *Melodic Comparison: Concepts, Procedure, and Applications*. Computing in Musicology 11 .
- Seltzer, M. (2002). Sphinx iii signal processing front end specification. [online]. Available <http://perso.enst.fr/sirocco/>, May 2002. [2006, 29 August]
- Smith, L.A. McNab, R.J. Witten I.H. (1998). Sequence-Based Melodic Comparison: A Dynamic-Programming Approach. Melodic Similarity. Concepts, Procedures, and Applications Computing in Musicology 11. 1001-117.
- Steiglitz, K. (1996). A Digital Signal Processing Primer: with Applications to Digital Audio and Computer Music. Addison-Wesley. New York. 22-43
- Smith, S. (1997). The Scientist and Engineer's Guide to Digital Signal Processing. California. California Technical Publishing. 35-66
- Shih, H. Narayanan, S.S. Jay Kou. C.C. (2003). Multidimension Humming Transcription Using Hidden Markov Models for Query by Humming Systems, Proceedings of the ISIMR 2003. [online]. Available [www.isimr.org](http://www.isimr.org) [2006, 16 February].
- Tidwell, J. (2005). Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly Media, Inc. 99-125.
- Typke, R. (2004). A Survey of Music Information Retrieval Systems, presented at the ISIMR 2004, Spain Available [www.isimr.org](http://www.isimr.org) [2006, 16 February].
- Tzanetakis, G. (no date). Marsyas User Manual. [online]. Available <http://www.sourceforge.marsyas.org> [2006, 16 February].
- Tzanetakis, G. Cook, P. (2000). Marsyas: A framework for audio analysis. Organized Sound, 4(3):169-175. Cambridge University Press
- Voorhees, E. (2004). Wither Music IR Evaluation infrastructure: Lessons to be learnt from TREC. The MIR/MDL Evaluation Project White Paper Collection, Establishing Music Information Retrieval (MIR) and Music Digital Library (MOL) Evaluation Frameworks: Preliminary Foundations and Infrastructures, 12-16.
- Walonick, D. (2004). Survival Statistics. StatPac, Inc. 15 -125.
- Witten, I. Frank, E. (2000). WEKA Machine learning algorithms in Java, Chapter 8 in Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers. [online]. Available <http://hartford.itlcs.cmu.edu/classes/95-779/HWlwekatutorial.pdf> [2006, 16 February].
- Yamamuro, Kosugi, Naoko & Nishihara, Yuichi & Sakata, Tetsuo &, Masashi & Kushima, Kazuhiko. (2002). A Practical Query-By-Humming System for a Large Music Database. In Proceedings ACM Multimedia: 333-

Zoia G., Zhou R., Mattavelli, M., (2002). MPEG Audio Coding and XML: samples, models, descriptors. [online]. Available <http://www.lim.dico.unimi.it/maxproject/max2002/docs/GZoiaMAX2002.pdf> [2006, 29 August].

Zhu, Y. Shasha, D. (2003). Query by humming: a time series database approach. In Proc. Of SIGMOD 2003. [online]. Available <http://citeseer.ist.psu.edu/zhu03query.html> [2006, 29 August].

(1998). UIUC DU Glossary. [online]. Available [dl.grainger.uiuc.edu/glossary.htm](http://dl.grainger.uiuc.edu/glossary.htm) [2006, 16 February].

(2005). MIREX 2005. [online]. Available [www.music-ir.org/mirexwiki/index.php/MIREX\\_2005](http://www.music-ir.org/mirexwiki/index.php/MIREX_2005) [2006, 16 February].

(2006). Wizard of Oz method. [online]. Available [www.usabilitynet.org/tools/wizard.htm](http://www.usabilitynet.org/tools/wizard.htm) [2006, 16 February].