Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

Master in Informatics Engineering

MSc. Thesis

Automatic Mood Tracking in Audio Music

Renato Eduardo Silva Panda

panda@student.dei.uc.pt

Thesis Supervisor: Professor Rui Pedro Paiva

July, 2010

Table of Contents

List of Figures

List of Tables

Glossary and Acronyms

Term	Description
ANN	Artificial Neural Network (Classifier)
AV	Arousal / Valence
BoostR	AdaBoost.RT (Regression Algorithm)
BPM	Beats Per Minute
CLI	Command Line Interface
DB	Database
DBMS	Database Management System
DWCH	Daubechies Wavelet Coefficient Histogram
FFS	Forward Feature Selection
FSA	Feature Selection Algorithm
GMM	Gaussian Mixture Model (Classifier)
k-NN	k-Nearest Neighbor (Classifier)
LPCC	Linear Prediction Cepstral Coefficients
LPGL	The GNU Lesser General Public License or LGPL is a free software license published by the Free Software Foundation (FSF).
LPRC	Linear Prediction Reflection Coefficients
LSP	Linear Spectral Pairs
MER	Music Emotion Recognition
MEVD	Music Emotion Variation Detection
MFCC	Mel-Frequency Cepstral Coefficient
MIR	Music Information Retrieval
MLR	Multiple Linear Regression (Regression Algorithm)
ODBC	Open Database Connectivity
PCA	Principal Component Analysis (method for reducing the correlation between Variables)
Precision	Ratio between the number of correctly detected boundaries and the total number of the detected boundaries (mood changes)
Qt	A cross-platform application and UI framework
Recall	Ratio between the number of correctly detected boundaries and that of the original boundaries (of mood changes in a music clip)
RMSE	Root Mean Square Error
RReliefF	A simple yet powerful feature selection algorithm
SCF	Spectral Crest Factor
SFM	Spectral Flatness Measure
SSE	Sum-Square Error
SST	Total Sum of Squares
STFT	Short Time Fourier Transformation
SVC	Support Vector Classification
SVM	Support Vector Machines, a set of related supervised learning methods used for classification and regression.
SVR	Support Vector Regression (Regression Algorithm)
UML	Unified Modeling Language

1. Introduction

"Music is what feelings sound like." – Author Unknown

Since the beginning of mankind music has always been present in our lives, serving a myriad of purposes both socially and individually. Used in fields as diverse as religion, sports, entertainment, health and even in war, *"music is what feelings sound like"* (Author Unknown), conveying emotions and perceptions to the listener, which vary between cultures and civilizations.

With the boom in technologies and the beginning of digital era, the music business started expanding and currently music is always present in our days, in our car, while working, exercising, in the streets, television and others. This frenetic growth in music supply and demand led to new distribution methods and nowadays every person has a large collection of albums and songs. These advances in technology exposed some of the current problems in music catalog and retrieval, an area that has been left behind, uncovering the need for powerful methods for automatically retrieving useful and relevant songs in a given context from such huge databases.

1.1. Motivation and Scope

Being involved in a scientific project is always exciting as it permits us to work on something cutting-edge, using state of the art technology, researching and planning our way towards the desired goals, uncovering new solutions to the posed problem. Since this is the final year of *MSc*, it can also be the last opportunity to be attached to the academic world and work on challenging problems, proposing solutions and researching to enrich the University legacy. To complement this, the project presented here is extremely interesting, focusing on music mood and bringing together two distinct areas: research and software engineering, being a tremendous challenge and motivation to me.

Since the last decade we have been assisting to an expansion of personal and public music collections but also the high growth in industry size and profits due to new distribution methods. It is expected that this frenetic growth will not invert or even slow down in the next years but increase even more as we move to a more global world each day. With it emerges the necessity of new tools and means, much more advanced and flexible than the ones existing now, making us capable of easily searching and browsing large music sets based on the needs of specific individuals. Music Information Retrieval (MIR) is a relatively new research area that is gaining more awareness due to these problems we face today, with several companies and research centers investing on this field. The goal is to find different ways of extracting useful information from data in songs, making searching, classification and indexing songs much faster and precise. These search criteria can be based on metadata information like title, artist, genre, query by example using a small audio excerpt, but also using criteria like mood, taste and emotion, that are subjective and may vary between individuals and cultures. Those subjective criteria will be approached in this work, since music mood analysis and tracking is still in the beginning and offers a wide range of new possibilities that are yet to explore. On the other hand, objective criteria analysis, where retrieving songs by author, title and genre as well as identifying and watermarking them are already common in the field.

Being able to query songs by example or searching based on emotion criteria opens enormous opportunities and offers a wide range of possible applications:

- Playlist generators and music selectors based on mood could give the possibility for users exercising or instructors to choose what kind of tracks they would like to listen to, ranging high tempo, fast songs to calm and relaxing songs, to be used in meditation sessions.
- Film directors could make use of these capacities to find songs that match the planned scene, instigating fear, anger, joy and other kind of emotions in the spectators.
- An advertiser on radio and television would have a great help when finding the right music to captivate desired clients.
- Call centers that tend to have clients waiting in line listening to the same classical music excerpt over and over could now automatically pick some more recent songs from alternative genres that would adjust to the objective of maintaining the costumer happily waiting.
- DJs passing music in parties looking for music with similar mood and beat so people will not notice changes between songs.
- Gaming industry, searching for the right sound to apply on specific moments to increase the tension, mark a moment of happiness, anger, revenge and other similar emotions frequently present in games.
- Any regular person who, after an exhausting day wants some relaxing music, songs that will cheer him up after some sad events.

1.2. Objectives and Approaches

The main objective of this work is to develop a platform able to identify mood and track mood variations during the entire audio musical clip. This tool should be capable of, among other functions, analyze and audio file and present the identified emotion (e.g., anger, depression, contentment, exuberance). To achieve this, a careful planning must be conducted before the software implementation, creating a detailed requirements analysis and software design, ensuring a stable and robust platform that is prepared to future improvement, since the project should not end with this work. In fact, it should be regarded rather as a starting point for future research and work.

Obviously, to develop a quality work in an area that is relatively new and unexplored means that a deep research in the field is imperative first, to gain adequate knowledge in the area and identify relevant information in several topics. Namely, audio features (upon which mood detection will strongly depend), mood classification methods, and ways to track mood variations over the music clip. This requires studying the research work done until know in the same area and documenting all this for future reference.

Another goal is to study the existent frameworks in the audio processing field, like Marsyas, which currently have some flaws in terms of manuals and documentation, discovering their capabilities and documenting a bit more on how they work and can be used.

Concluding, the project approaches two distinct areas, having aspects and phases of both a research project and a software engineering project. The research part occupied the first semester, where the state of the art in MIR was studied and know-how on existent approaches, algorithms, classifiers, tools and frameworks was acquired. This was essential to produce a quality planning. The research part continued in the second semester, along with the software engineering part, with algorithm analysis and evaluation.

The software engineering followed a waterfall development model, going through all necessary phases to implement the desired tool. Based on research and following especially Yang et *al.*'s approach [Yang et al., 2008], a mood tracking system was developed, testing features available in the Marsyas framework and evaluating the performance of the most recommended classifier [Yang et al., 2008], and Support Vector Machines (SVM).

After ending the initial phases of identifying requirements, planning the architecture and all the software design phase, the implementation part took place. The application was developed in the C++ language using the Marsyas framework, which has proven to be robust and very fast when comparing with the existent alternatives (MIREX 2008 results[1]). The Qt UI framework was employed to create graphical user interfaces since it is portable and integrates well with Marsyas, besides its generalized use.

The planned application followed a client-server architecture, where the server is responsible for handling client connections and process audio files to create the music mood database. Clients, on the other side, are able to add new songs to the database and also to browse this information, viewing mood changes in songs by retrieving a previously analyzed song.

Part of this work was developed jointly by myself and João Fernandes, who is also working in his MSc thesis, titled "Automatic Playlist Generation via Music Mood Analysis". Our objective is similar and some of our work was done together, namely regarding the planning phases of the application. However, the objectives take different paths in some parts. While I will be focused in mood tracking, providing graphical feedback on how and where the mood

[1] http://www.music-ir.org/mirex/2008/index.php/Audio_Music_Mood_Classification_Results

changes during an entire song, João's work is focused on querying new songs by example and playlist generation based on a given music or a point in the mood map.

Finally, sets of tests were conducted, from an engineering point of view, testing the developed software and fixing identified bugs. . Moreover, model validation was also carried out , using as base a test collection created and previously used in [Yang et al., 2008] as well as additional annotations intended to mood tracking (performed by our team).

1.3. Work Plan

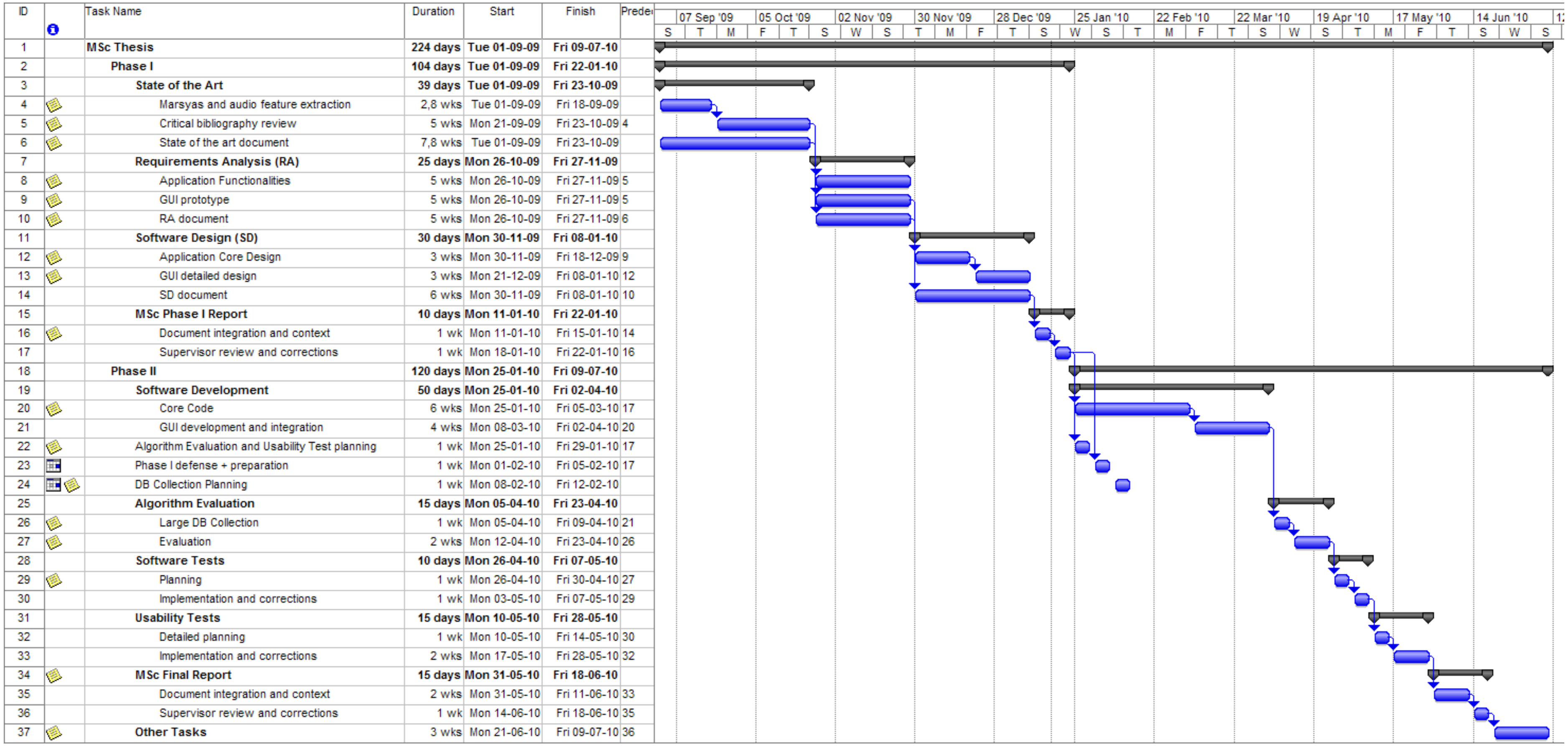
The thesis had the duration of one scholar year with the work being divided in two parts, one in each semester, with a report being produced at the end of each one. The first semester consisted mainly on theoretical work, with the study and review of relevant MIR topics that gave us a good knowledge of the area and help us in the planning of the software to develop. The main tasks were:

- Study and summarization of the state of the art.
- Requirements analysis.
- Review of mood based feature extraction.
- Software design (with graphical user interface tests).
- Documentation and intermediate report.

The second semester was focused on implementation. Starting from the foundations and knowledge built before and following software engineering processes the planned software was implemented. Using the developed tools, various experiments were conducted in order to evaluate mood detection and tracking results using a variety of settings and approaches. A prototype version of the server and client applications was also produced, demonstrating the future possibilities for the project. We have also conducted software tests in order to guarantee the robustness of the tool, having implemented the needed fixes and refinements.

The detailed planning of the project is visible at three different stages on the Figure 1 (initial plan), Figure 2 (end of phase 1) and Figure 3 (end of phase 2).

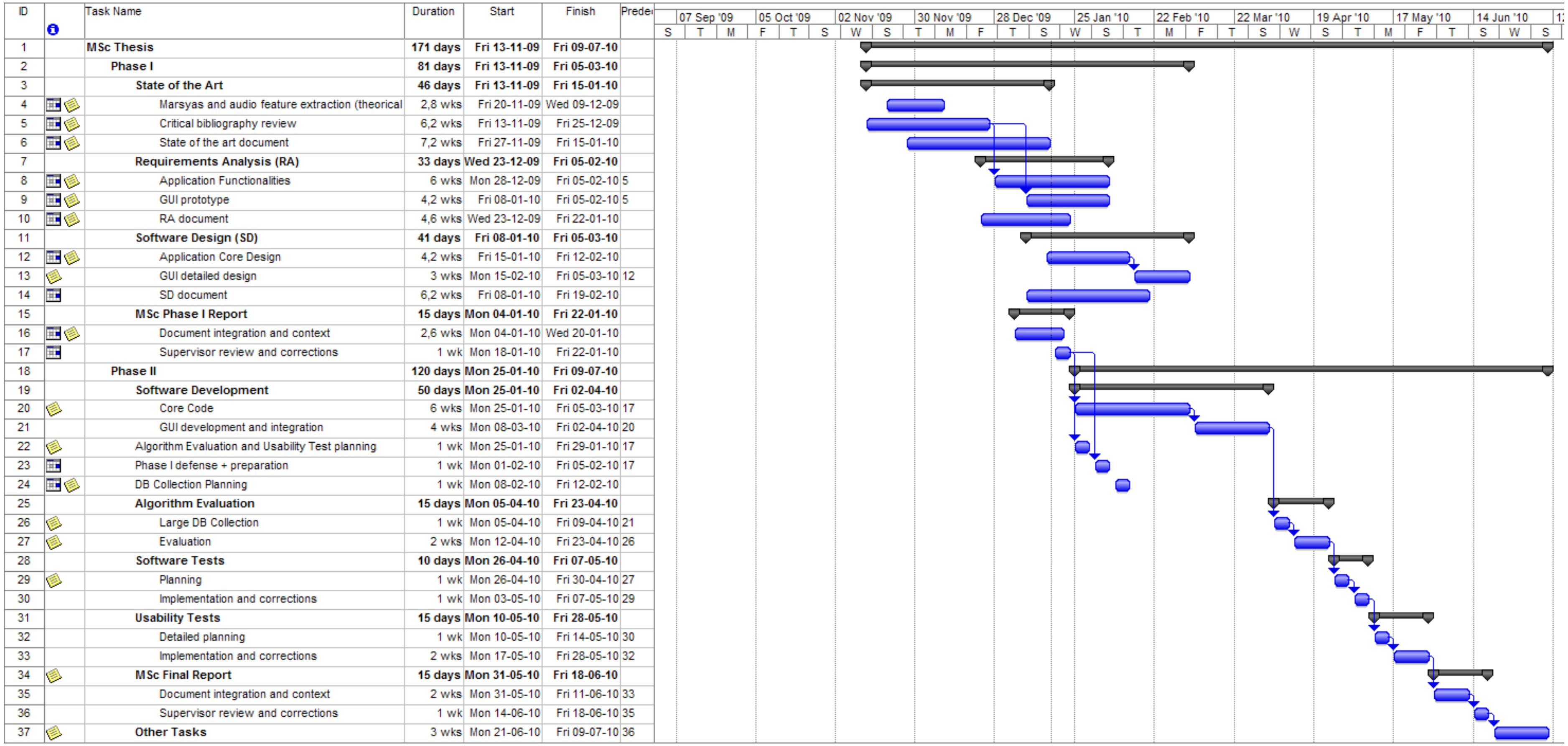
Gantt Diagram – Initial Plan



ID		Task Name	Duration	Start	Finish	Prede
1		**MSc Thesis**	**224 days**	**Tue 01-09-09**	**Fri 09-07-10**	
2		**Phase I**	**104 days**	**Tue 01-09-09**	**Fri 22-01-10**	
3		**State of the Art**	**39 days**	**Tue 01-09-09**	**Fri 23-10-09**	
4		Marsyas and audio feature extraction	2,8 wks	Tue 01-09-09	Fri 18-09-09	
5		Critical bibliography review	5 wks	Mon 21-09-09	Fri 23-10-09	4
6		State of the art document	7,8 wks	Tue 01-09-09	Fri 23-10-09	
7		**Requirements Analysis (RA)**	**25 days**	**Mon 26-10-09**	**Fri 27-11-09**	
8		Application Functionalities	5 wks	Mon 26-10-09	Fri 27-11-09	5
9		GUI prototype	5 wks	Mon 26-10-09	Fri 27-11-09	5
10		RA document	5 wks	Mon 26-10-09	Fri 27-11-09	6
11		**Software Design (SD)**	**30 days**	**Mon 30-11-09**	**Fri 08-01-10**	
12		Application Core Design	3 wks	Mon 30-11-09	Fri 18-12-09	9
13		GUI detailed design	3 wks	Mon 21-12-09	Fri 08-01-10	12
14		SD document	6 wks	Mon 30-11-09	Fri 08-01-10	10
15		**MSc Phase I Report**	**10 days**	**Mon 11-01-10**	**Fri 22-01-10**	
16		Document integration and context	1 wk	Mon 11-01-10	Fri 15-01-10	14
17		Supervisor review and corrections	1 wk	Mon 18-01-10	Fri 22-01-10	16
18		**Phase II**	**120 days**	**Mon 25-01-10**	**Fri 09-07-10**	
19		**Software Development**	**50 days**	**Mon 25-01-10**	**Fri 02-04-10**	
20		Core Code	6 wks	Mon 25-01-10	Fri 05-03-10	17
21		GUI development and integration	4 wks	Mon 08-03-10	Fri 02-04-10	20
22		Algorithm Evaluation and Usability Test planning	1 wk	Mon 25-01-10	Fri 29-01-10	17
23		Phase I defense + preparation	1 wk	Mon 01-02-10	Fri 05-02-10	17
24		DB Collection Planning	1 wk	Mon 08-02-10	Fri 12-02-10	
25		**Algorithm Evaluation**	**15 days**	**Mon 05-04-10**	**Fri 23-04-10**	
26		Large DB Collection	1 wk	Mon 05-04-10	Fri 09-04-10	21
27		Evaluation	2 wks	Mon 12-04-10	Fri 23-04-10	26
28		**Software Tests**	**10 days**	**Mon 26-04-10**	**Fri 07-05-10**	
29		Planning	1 wk	Mon 26-04-10	Fri 30-04-10	27
30		Implementation and corrections	1 wk	Mon 03-05-10	Fri 07-05-10	29
31		**Usability Tests**	**15 days**	**Mon 10-05-10**	**Fri 28-05-10**	
32		Detailed planning	1 wk	Mon 10-05-10	Fri 14-05-10	30
33		Implementation and corrections	2 wks	Mon 17-05-10	Fri 28-05-10	32
34		**MSc Final Report**	**15 days**	**Mon 31-05-10**	**Fri 18-06-10**	
35		Document integration and context	2 wks	Mon 31-05-10	Fri 11-06-10	33
36		Supervisor review and corrections	1 wk	Mon 14-06-10	Fri 18-06-10	35
37		**Other Tasks**	3 wks	Mon 21-06-10	Fri 09-07-10	36

Figure 1: Gantt diagram – Initial Plan

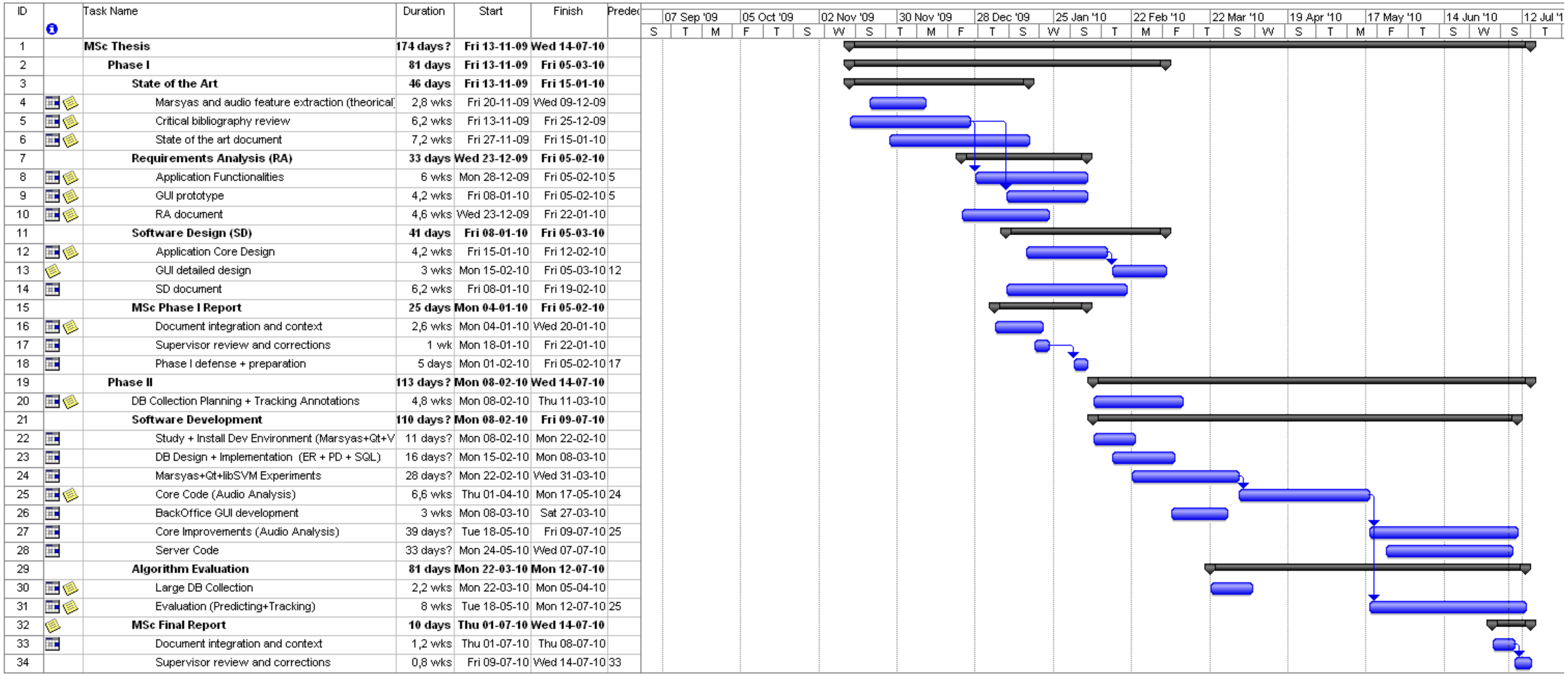
Gantt Diagram – End of Phase 1



ID	ℹ	Task Name	Duration	Start	Finish	Prede
1		**MSc Thesis**	**171 days**	**Fri 13-11-09**	**Fri 09-07-10**	
2		**Phase I**	**81 days**	**Fri 13-11-09**	**Fri 05-03-10**	
3		**State of the Art**	**46 days**	**Fri 13-11-09**	**Fri 15-01-10**	
4	📅 📝	Marsyas and audio feature extraction (theorical	2,8 wks	Fri 20-11-09	Wed 09-12-09	
5	📅 📝	Critical bibliography review	6,2 wks	Fri 13-11-09	Fri 25-12-09	
6	📅 📝	State of the art document	7,2 wks	Fri 27-11-09	Fri 15-01-10	
7		**Requirements Analysis (RA)**	**33 days**	**Wed 23-12-09**	**Fri 05-02-10**	
8	📅 📝	Application Functionalities	6 wks	Mon 28-12-09	Fri 05-02-10	5
9	📅 📝	GUI prototype	4,2 wks	Fri 08-01-10	Fri 05-02-10	5
10	📅 📝	RA document	4,6 wks	Wed 23-12-09	Fri 22-01-10	
11		**Software Design (SD)**	**41 days**	**Fri 08-01-10**	**Fri 05-03-10**	
12	📅 📝	Application Core Design	4,2 wks	Fri 15-01-10	Fri 12-02-10	
13	📝	GUI detailed design	3 wks	Mon 15-02-10	Fri 05-03-10	12
14	📅	SD document	6,2 wks	Fri 08-01-10	Fri 19-02-10	
15		**MSc Phase I Report**	**15 days**	**Mon 04-01-10**	**Fri 22-01-10**	
16	📅 📝	Document integration and context	2,6 wks	Mon 04-01-10	Wed 20-01-10	
17	📅	Supervisor review and corrections	1 wk	Mon 18-01-10	Fri 22-01-10	
18		**Phase II**	**120 days**	**Mon 25-01-10**	**Fri 09-07-10**	
19		**Software Development**	**50 days**	**Mon 25-01-10**	**Fri 02-04-10**	
20	📝	Core Code	6 wks	Mon 25-01-10	Fri 05-03-10	17
21		GUI development and integration	4 wks	Mon 08-03-10	Fri 02-04-10	20
22	📝	Algorithm Evaluation and Usability Test planning	1 wk	Mon 25-01-10	Fri 29-01-10	17
23	📅	Phase I defense + preparation	1 wk	Mon 01-02-10	Fri 05-02-10	17
24	📅 📝	DB Collection Planning	1 wk	Mon 08-02-10	Fri 12-02-10	
25		**Algorithm Evaluation**	**15 days**	**Mon 05-04-10**	**Fri 23-04-10**	
26	📝	Large DB Collection	1 wk	Mon 05-04-10	Fri 09-04-10	21
27	📝	Evaluation	2 wks	Mon 12-04-10	Fri 23-04-10	26
28		**Software Tests**	**10 days**	**Mon 26-04-10**	**Fri 07-05-10**	
29	📝	Planning	1 wk	Mon 26-04-10	Fri 30-04-10	27
30		Implementation and corrections	1 wk	Mon 03-05-10	Fri 07-05-10	29
31		**Usability Tests**	**15 days**	**Mon 10-05-10**	**Fri 28-05-10**	
32		Detailed planning	1 wk	Mon 10-05-10	Fri 14-05-10	30
33		Implementation and corrections	2 wks	Mon 17-05-10	Fri 28-05-10	32
34	📝	**MSc Final Report**	**15 days**	**Mon 31-05-10**	**Fri 18-06-10**	
35		Document integration and context	2 wks	Mon 31-05-10	Fri 11-06-10	33
36		Supervisor review and corrections	1 wk	Mon 14-06-10	Fri 18-06-10	35
37	📝	**Other Tasks**	3 wks	Mon 21-06-10	Fri 09-07-10	36

Figure 2: Gantt diagram – End of Phase 1

Gantt Diagram – End of Phase 2



ID	Task Name	Duration	Start	Finish	Predec
1	**MSc Thesis**	**174 days?**	**Fri 13-11-09**	**Wed 14-07-10**	
2	**Phase I**	**81 days**	**Fri 13-11-09**	**Fri 05-03-10**	
3	**State of the Art**	**46 days**	**Fri 13-11-09**	**Fri 15-01-10**	
4	Marsyas and audio feature extraction (theorical	2,8 wks	Fri 20-11-09	Wed 09-12-09	
5	Critical bibliography review	6,2 wks	Fri 13-11-09	Fri 25-12-09	
6	State of the art document	7,2 wks	Fri 27-11-09	Fri 15-01-10	
7	**Requirements Analysis (RA)**	**33 days**	**Wed 23-12-09**	**Fri 05-02-10**	
8	Application Functionalities	6 wks	Mon 28-12-09	Fri 05-02-10	5
9	GUI prototype	4,2 wks	Fri 08-01-10	Fri 05-02-10	5
10	RA document	4,6 wks	Wed 23-12-09	Fri 22-01-10	
11	**Software Design (SD)**	**41 days**	**Fri 08-01-10**	**Fri 05-03-10**	
12	Application Core Design	4,2 wks	Fri 15-01-10	Fri 12-02-10	
13	GUI detailed design	3 wks	Mon 15-02-10	Fri 05-03-10	12
14	SD document	6,2 wks	Fri 08-01-10	Fri 19-02-10	
15	**MSc Phase I Report**	**25 days**	**Mon 04-01-10**	**Fri 05-02-10**	
16	Document integration and context	2,6 wks	Mon 04-01-10	Wed 20-01-10	
17	Supervisor review and corrections	1 wk	Mon 18-01-10	Fri 22-01-10	
18	Phase I defense + preparation	5 days	Mon 01-02-10	Fri 05-02-10	17
19	**Phase II**	**113 days?**	**Mon 08-02-10**	**Wed 14-07-10**	
20	DB Collection Planning + Tracking Annotations	4,8 wks	Mon 08-02-10	Thu 11-03-10	
21	**Software Development**	**110 days?**	**Mon 08-02-10**	**Fri 09-07-10**	
22	Study + Install Dev Environment (Marsyas+Qt+V	11 days?	Mon 08-02-10	Mon 22-02-10	
23	DB Design + Implementation (ER + PD + SQL)	16 days?	Mon 15-02-10	Mon 08-03-10	
24	Marsyas+Qt+libSVM Experiments	28 days?	Mon 22-02-10	Wed 31-03-10	
25	Core Code (Audio Analysis)	6,6 wks	Thu 01-04-10	Mon 17-05-10	24
26	BackOffice GUI development	3 wks	Mon 08-03-10	Sat 27-03-10	
27	Core Improvements (Audio Analysis)	39 days?	Tue 18-05-10	Fri 09-07-10	25
28	Server Code	33 days?	Mon 24-05-10	Wed 07-07-10	
29	**Algorithm Evaluation**	**81 days**	**Mon 22-03-10**	**Mon 12-07-10**	
30	Large DB Collection	2,2 wks	Mon 22-03-10	Mon 05-04-10	
31	Evaluation (Predicting+Tracking)	8 wks	Tue 18-05-10	Mon 12-07-10	25
32	**MSc Final Report**	**10 days**	**Thu 01-07-10**	**Wed 14-07-10**	
33	Document integration and context	1,2 wks	Thu 01-07-10	Thu 08-07-10	
34	Supervisor review and corrections	0,8 wks	Fri 09-07-10	Wed 14-07-10	33

Figure 3: Gantt diagram – End of Phase 2

2. Mood Tracking in Audio Music: Context and Overview

2.1. Mood and Emotion

Definition of Emotion

Emotion is a complex set of interactions among subjective and objective factors, mediated by neural/hormonal systems, which can (a) give rise to affective experiences such as feelings of arousal, pleasure/displeasure; (b) generate cognitive processes such as perceptually relevant effects, appraisals, labeling processes; (c) activate widespread physiological adjustments to the arousing conditions; and (d) lead to behavior that is often, but not always, expressive, goal-oriented, and adaptive [Kleinginna and Kleinginna, 1981; cited in Meyers, 2007].

Definition of Mood

A mood is a relatively long lasting emotional state. Moods differ from simple emotions in that they are less specific, less intense, and less likely to be triggered by a particular stimulus or event [Thayer, 1989].

For a very long time, mood and emotions have been a major subject of psychologists, being analyzed and discussed often to create the best model to represent emotions. Mood and emotions are subjective, varying from person to person and also across cultures. Furthermore usually there are many words describing emotions, with some being direct synonyms while others representing small variations. Different persons have different perceptions of the same stimulus and often use some of these different words to describe similar experiences. Unfortunately, there is not a standard, widely accepted, mood taxonomy. That said, correctly studying and understanding the existent models for representing emotions, and wisely choosing the one that better fits our needs can be seen has one important foundation of this work. In order to be useful in music emotion recognition (MER) there are a few main qualities that models need to cover:

- Obviously, it has to accurately represent reality.
- The most common emotions existent in music must be present in model.
- The model should have one or more dimensions in order to measure emotions.

Several theoretical models have been proposed over the years by authors from *Hevner* to *Thayer*. These models can be grouped in two major approaches: categorical models or dimensional models. Categorical models consist of several categories or states of emotion, such as anger, fear, happiness and joy, one example of this is Hevner's adjective circle. Dimensional models, on the other hand, use several axes to map emotions to a plan. The most frequent approach is using two axes (e.g. arousal-valence (AV) or energy-stress), with some cases of a third dimension (dominance).

The benefit of dimensional models is the reduced ambiguity when compared with the categorical approach. However, some ambiguity still exists, since each of the four quadrants can represent more than one distinct emotion (happiness and excitation are both represented by high arousal and valence for example). Given this, dimensional models can be further divided into discrete (representing the ones described above) and continuous. Continuous models, unlike discrete ones, view the emotion plan as a continuous space where each point denotes a different emotional state. As a result, all ambiguity related with emotion states is removed, however the issue related with arousal and valence dependency still remains. Thayer's model of mood can fit in both sub-categories: it can be considered discrete, having four classes, but it can also be regarded as a continuous model, as approached by [Yang et al., 2008].

Next, we will introduce some of the most interesting mood models.

Hevner's Adjective Circle

Kate Hevner is best known by her research in music psychology, being one of the first to do research on the subject of music mood [Hevner, 1936]. She concluded that music and emotions are intimately connected, with music always carrying emotional meaning in it. As a result, she introduced an emotion (adjective) list, known has Hevner's Adjective Circle (Figure 4). This model was later used as a base for some subsequent research in the area.

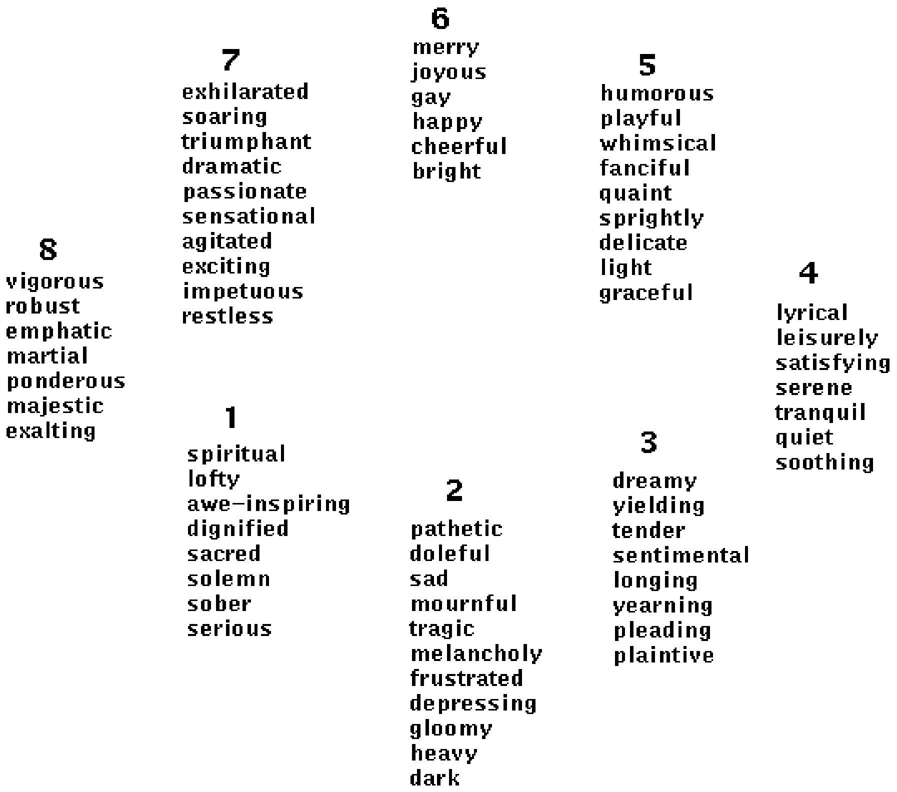


Figure 4: Hevner's adjective circle [Meyers, 2007]

Hevner's list is composed by 67 different adjectives such as serene, tranquil, and quiet which are organized in eight different groups in a circular way. However, these list features a high number of different emotions in each cluster, most of them being of similar or very close meaning. This ambiguity raises several difficulties in discriminating one from another to obtain the "ground truth".

Thayer's Model of Mood

In 1989 Robert Thayer proposed a two-dimensional mood model [Thayer, 1989], offering a simple but effective way to represent mood. Thayer adopted a distinct approach than Hevner's adjectives list, stating that mood depends on two factors: Stress (happiness/anxiety) and Energy (calm/energy) combined in a two-dimensional axis forming four different quadrants: Contentment, representing calm and happy music; Depression, referring to calm and anxious music; Exuberance, referring to happy and energetic; and Anxiety, representing frantic and energetic music (see in Figure 5). One of the less strong aspects of this model is its low granularity, not having a high number of well defined different emotions. On the other hand, its simplicity is an advantage, making it less ambiguous. Finally, a key aspect of the model is that emotions are situated far away from the center, due to the fact that the center represents the origin of the referential, where both arousal and valence have small values, not representing a clear, identifiable emotion.

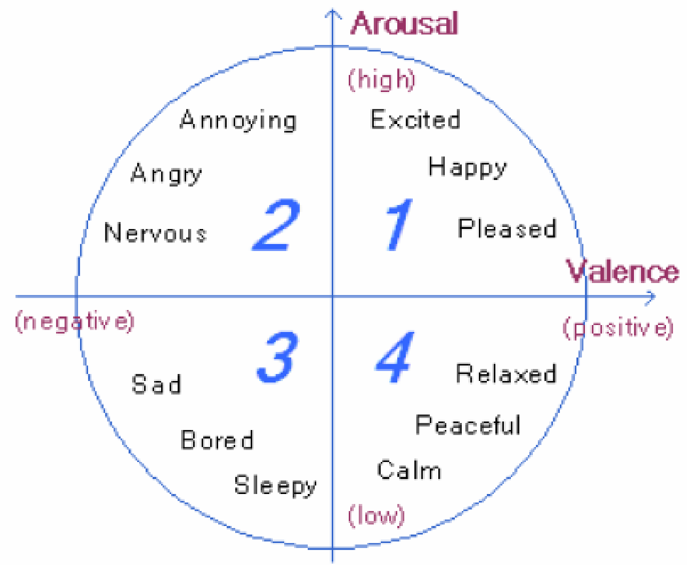


Figure 5: Thayer's arousal-valence emotion plane [Yang et al., 2008]

Tellegen-Watson-Clark Model of Mood

Tellegen-Watson-Clark model of mood [Tellegen et al., 1999] is a model proposed by the three researchers (Tellegen, Watson and Clark), containing a high number of emotions, in contrast to Thayer's model, organized in a circular way like Hevner's model, using the positive/negative affect as one dimension and the pleasantness/unpleasantness versus engagement/disengagement (45 degrees rotated) as the other.

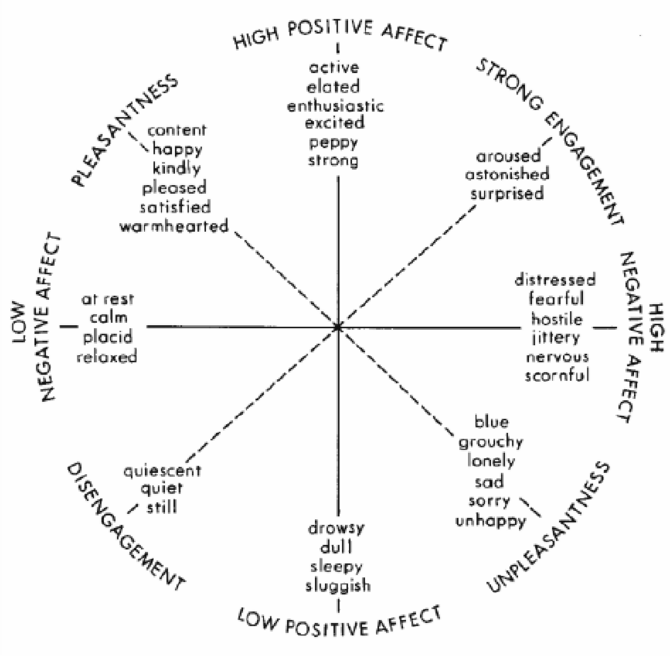


Figure 6: Tellegen-Watson-Clark model of mood [Laar, 2006]

Naturally, although more models and taxonomies exist and are subject of study we have only approached three of the most relevant to this work. Names like Farnsworth [Li et al., 2003] and Russell [Meyers, 2007] are examples of it.

Comparison of Mood Models

A brief comparison between the three presented methods and a summary description of each one is presented in Table 1.

Model of Mood	Perspective	Granularity (clusters/emotions)	Description
[Hevner, 1936]	Categorical	8 / 67	List of adjectives /emotions organized in eight clusters
[Thayer, 1989]	Continuous	4 / 12	Two-dimensional representation with two axis using values for Energy and Stress in each one, dividing space in four quadrants.
[Tellegen, Watson, & Clark, 1999]	Continuous	8 / 38	Circular representation using positive / negative affect as one dimension and pleasantness / unpleasantness versus engagement / disengagement (45° rotated) as the other.

Table 1: Comparison of Mood Models

2.2. Mood Analysis in MIR Research

With the increasing interest in having ways to identify and extract valuable data from audio music with good accuracy, giving us a wide range of opportunities in music databases search and categorization, we have assisted in this last decade to the appearance of new research in areas like Music Information Retrieval (MIR), Music Emotion Recognition (MER) and Music Emotion Variation Detection (MEVD) addressing these needs.

In this section we will analyze some of the most interesting and promising publications that are relevant to this thesis.

Emotion Detection in Music, a Survey [Laar, 2006]

In this paper the author presents a comparison between some publications with different methods of emotion detection, making the distinction between the accuracy of each method, the granularity of mood taxonomies used and some possible applications.

It starts by giving a brief view over some audio features in music that were used in the analyzed papers, organizing them into eight distinct groups:

- Musical surface (or timbre texture) composed with features mostly based on the Short-Time Fourier Transformation (STFT). Some of those features are: centroid, roll off, spectral flux, zero crossings and low energy / average silence ratio.
- Spectral Flatness Measure (also called tonality coefficient).
- Spectral Crest Factor.
- Mel Frequency Cepstral Coefficients (MFCC) (often used in speech recognition).
- Daubechies Wavelet Coefficient Histogram (DWCH).
- Beat and tempo detection.
- Genre information, which tends to be expensive, needing to be handcrafted in songs or pooled from internet. It is also prone to errors.
- Lyrics.

After the features introduction, three models of mood are referred to: Hevner’s list, Thayer’s model and Tellegen-Watson-Clark’s model, then discussing some issues in MER methods, ranging from precision/granularity/diversity (related to the mood model used) to questions about computer power, appropriated learning algorithms and the cultural background that makes emotion detection so subjective in different world regions.

Finally the article presents six emotion detection methods from several authors (between 2003 and 2005) with different approaches, comparing them based on accuracy, granularity, diversity and selection.

Sentiment Retrieval in Popular Music based on Sequential Learning [Carvalho et al., 2005]

In this article the authors propose a new taxonomy, using a five-point scale based on the "happiness" present in a song, that in their view will result in more appealing labels to the users in future applications and also will drop the complexity of the algorithm that would exist if more categories and adjectives were used.

The test collection used to evaluate the results consists in 200 songs that were previously classified by two persons with an agreement between the two of 82%.

Songs are converted to 22050Hz, 16 bit mono and four classes of musical features are used: Musical Surface, Spectral Flatness Measure, Spectral Crest Factor and Mel Frequency Cepstral Coefficients. The classification is done using two different learning algorithms and classifiers.

The conclusion reached by the study shows that taxonomy granularity, using a binary problem against a more "fine-grained" problem (of five labels) has a much higher impact on precision (13.5% vs. 63.45% in error rate) than the used classifiers and learning algorithms, which only made de results vary within 63.45% and 67%.

Some of the problems and limitations of this method are related with the absence of information on learning algorithms and also the lack of details in the test collection as well as the features used.

Detecting Emotion in Music [Li et al., 2003]

The main objective of this work is to develop a classification system for music, admitting that the same song can have more than one emotion during the entire clip.

The taxonomy used consists of the ten emotions present in the Farnsworth model and three extra emotions added according to a test subject who indexed the test songs. The musical database for this test was composed of 499 songs, with 50% being used for training and the remaining 50% used for testing. These songs were selected from 128 music albums (at least four songs each) and the collection covered four major music types: Ambient (120 files), Classical (164 files), Fusion (135 files), and Jazz (100 files).

The employed acoustic features used consisted on timbral texture features, rhythm content features (beat and tempo detection) and pitch content features, with the used classifiers being based on Support Vector Machines.

The final results were modest, with an accuracy of around 50%, still better than the last paper, while a higher granularity was used here.

Content-based Music Similarity Search and Emotion Detection [Li et al., 2004]

This paper describes a system that addresses two different objectives: similarity search, returning results that are (more or less) similar to the selected musical piece and also mood detection in a song.

The mood taxonomy is of low granularity, using the following categories: (Cheerful, Depressing), (Relaxing, Exciting) and (Comforting, Disturbing). As for the test collection used, it is composed of 235 musical pieces, all from the same genre – Jazz. Two subjects were responsible for the annotations. The extracted audio features are Mel Frequency Cepstral Coefficients, Musical surface features and Daubechies Wavelet Filters, which, according to the author, returned a total of 35 different frequency bands but not all were relevant. No detailed information on the used classifiers was presented.

The results showed a good accuracy, ranging from 70% to 83%. However it should be noted that the granularity was low and only one musical genre was tested. Accuracy results also varied between the two subjects, possibly due to different cultural backgrounds. One suggestion to solve this is to let the algorithms be trained for each subject. It would also be interesting to see how it would perform with different musical genres.

Music Information Retrieval by Detecting Mood via Computational Media Aesthetics [Feng et al., 2003]

In this research a new system for mood detection in music is present, stating that tempo and articulation in a music piece can be used to identify emotions. Only four emotions are used in this system: Happiness, Sadness, Anger and Fear. To extract tempo, the authors states that it is possible to use more than one algorithm. Determining the type of articulation is done using Average Silence Ratio, which represents what percentage of sound in one frame is below the average level. The classification is then performed recurring to a neural network with three layers, deciding the music score for each one of the four categories.

The tests were carried out using a collection of 330 songs for training but only 23 were used for testing purposes. This seriously compromises the results since the sample is too low and could even be handpicked to generate good results. Other problems related with the test collection are the lack of information about the musical genres or details on how it was annotated.

The final results are also awkward, with three of the four categories (Happiness, Sadness and Anger) scoring high results, between 75% and 86%, and fear only reaching 25%. This is, in part, to the restricted test collection, where only three songs were identified with Fear emotion.

Automatic Mood Detection from Acoustic Music Data [Liu et al., 2003]

The main focus of this paper is on mood detection in classical music and mood tracking, taking into consideration the fact that mood can change along the music, especially classical music. To achieve this, the Thayer's model of mood serves as base for the algorithm, with the intensity of the music being mapped to energy and both timbre and rhythm mapped to the stress component. The used algorithms rely on features such as Root Mean Square value in each sub band (for intensity), spectral shape features like Centroid, Roll off and Spectral flux (for timbre) and a Canny estimator, used to detect beat (for rhythm). The resulting information is then processed by two frameworks with distinct approaches, hierarchical and non-hierarchical. The hierarchical approach is conducted in two steps: it first splits songs in two groups based on intensity (Depression / Contentment versus Exuberance / Anxious); then, the second step makes the distinction between the two emotions existent in the selected group. In the non-hierarchical framework all the features are applied in a single step, making immediately the choice between one of the existent four groups.

The database used for evaluation consists of 250 classical music pieces, split into 20 seconds clips, 75% being used for training and 25% for testing. The results are very good, with accuracy reaching values from 76.6% to 94.5% for the hierarchical framework, with the non-hierarchical reaching 64.7% to 94.2%. Although the results were in fact very high, the fact that only classical music was used and the low granularity of the mood taxonomy, with only four possible emotions should be taken into consideration.

Disambiguating Music Emotion Using Software Agents [Yang et al., 2004]

This work tries to extract mood information from music, comparing the results with human annotations for the same music. A different approach is taken in this paper, with annotated lyrics being also analyzed. The chosen mood model is the Tellegen-Watson-Clark's, with authors giving more emphasis to the more negative emotions since, according to the article, they are harder to identify. Wavelet tools, beats per minute (BPM) detection methods and timbral features together with 12 features from Sony's EDS system [Pachet et al., 2004; cited in Yang et al., 2004] were used. The results showed a correlation of 0.90 between annotated and algorithm values, with features like BPM detection and Sum of Absolute Values of the Normalized FFT performing the best in identifying songs according to the two existent groups. Following this, lyrics are analyzed for defined keywords in order to differentiate between emotions.

The database used is composed by 152 Alternative Rock music pieces, 30 seconds each, with lyrics being present in 145 of them. The analysis of these lyrics helped differentiating songs that were in the same category, with an accuracy of 82.8%.

Automatic Mood Detection and Tracking of Music Audio Signals [Lu et al., 2006]

This paper is about automatic music mood detection on acoustic music data and also mood tracking on a music piece, by dividing the music into several independent segments, each containing a homogenous emotional expression. From the same authors of one of the previous papers [Liu et al., 2003], this work improves and continues the methods proposed there.

Using Thayer's mood taxonomy to classify emotions, the authors approach distinct solutions by proposing two types of frameworks, hierarchical and non-hierarchical, showing the advantages of using the first and being capable of emphasizing the most suitable features to different detection tasks. Both frameworks classify music sets based on the following feature sets (first each clip is down-sampled to 16 kHz, 16 bits, mono channel and divided into non overlapping frames of 32ms length): 1 – intensity (energy in each sub band), 2 – timbre, composed by MFCC (complemented with octave-based spectral contrast), spectral shape features (brightness, bandwidth, roll off and spectral flux) and spectral contrast features (sub-band peak, sub-band valley and sub-band contrast); 3 - rhythm (rhythm strength, rhythm regularity and tempo).

The non-hierarchical framework uses a single GMM combining the four mood clusters. It receives results from all extracted features from the three sets (intensity, timbre and rhythm), returning the calculated results. On the other hand, the hierarchical framework is a bit more complex, using several GMM with 16 mixture models. Each GMM is built using a set of features regarding each mood cluster, organized in three layers. As an example, a song classified in GMM_1 as having low intensity will be either contentment or depression, descending to layer two it will be classified by GMM_2 (using timbre features) and GMM_3 (using rhythm). The results are summed and one of the two moods is chosen (contentment or depression). Even though more complex, the hierarchical framework gives better results and makes a better use of sparse training data, important when the available training data is limited.

As for mood tracking, the goal was to go away from some previous ideas that used a sliding window of certain length to identify mood in that piece. However, such approach would most of the times have mixed moods contained in the same windows and thus could not be recognized correctly. The new approach proposed by authors tries to find potential mood change boundaries. This method consists in a two step mood tracking scheme. First, the goal is to find potential boundaries, recurring to intensity (using the intensity outline to detect possible boundaries), timbre and rhythm (to check for possible mood changes in possible boundaries). Then, the musical clip is divided into several independent segments, each containing a constant mood.

Experiments were conducted using a music database of 250 music pieces, mostly classical, which can be seen has a potential problem since the algorithms can be tuned to this specific genre only. These music pieces were annotated by music experts according to the four mood clusters, ignoring clips where there was no consensus about the existent mood. From that 75% of the clips were used for training and the remaining 25% for testing.

The results showed an average precision on mood detection of 86.3%, with average recall of 84.1%. In mood tracking tests the results showed that about 84.1% of the boundaries are recalled and the precision is about 81.5%. The results however, have much room for improvements, not only by finding more powerful audio features but by using various music genres and having more mood types covered in the used taxonomy.

A Mood-Based Music Classification and Exploration System [Meyers, 2007]

In his MSc thesis, Owen Craigie Meyers proposed to design and develop a tool that would allow users to automatically generate a playlist that suits a desired activity or mood, using audio information extracted from songs with context-aware data such as song lyrics. The author starts by presenting the background of MIR and psychology of music in general. After that, several fields are approached, e.g., the existent emotion models, feature extraction and classification frameworks, playlist generation tools and natural language processing. There is also a focus on the most popular music recommendation systems and how they could be useful to evaluate the final work.

Following this research, the author made several choices based on it to design and implement the application, namely an updated version of Hevner's emotions model adjective list by Schubert was mapped to Russell's model. Audio analysis was done using the CLAM[2] framework to extract mode and harmony, while tempo, rhythm and loudness are extracted with ENCLIAnalyzer[3]. The lyrics processing is done using Lyricator[4] to search and download the appropriate song lyric, that is then passed to a modified version of guess_mood function included in ConceptNet's natural language tools, outputting emotional concepts based on Russell's emotional model. The final steps are related with the song classification, which is done using at first a decision tree (for preliminary classification of the song database and then applying a k-nearest neighbor (k-NN) classification algorithm. This result is later combined with the lyric's affective value to give the global emotion for the song.

The evaluation of the system performance and lyrics classification was done using a database of 372 songs. The results obtained were compared with three different sources:

[2] http://clam-project.org/

[3] http://www.echonest.com/

[4] http://www.media.mit.edu/_meyers/lyricator.php

Experts evaluation, from All Music Guide[5] and other projects like Pandora Internet Radio[6] and The Music Genome Project; using social tagging networks like Last.FM[7], Qloud[8] and MyStrands; and user evaluation, based on a study where 12 students tested the program. In general, the results were good, with the system being able to correctly identify a good amount of songs (although the author did not properly quantified it). A few weaknesses and problems emerged, mainly in the feature extraction stage, due to the usage of only five features, losing useful information (pitch, melody, timbre) with this decision. Also, misclassifications in some features like tempo or mode had a large impact on the classification. In lyrics analysis, there were some problems too due to the lack of any sense of song-level semantic content, resulting in misclassifications. Nevertheless lyrics proved to be of great value to give a correct identification of moods.

As a final note some future improvements were listed, being the most relevant the addition of newer, more complex and powerful audio features, the usage of regression analysis to make the system more statistically objective and increasing the classification results by improving the current classification algorithms or using alternatives like Support Vector Machines (SVM) and Neural Networks.

A Regression Approach to Music Emotion Recognition [Yang et al., 2008]

This paper presents a solution to music emotion recognition (MER). To this end, the author follows a regression approach, trying to track mood by predicting the arousal and valence values of each music sample. To categorize these results, the Thayer's model of mood, consisting of the continuous (not discrete) arousal/valence emotion plane, is used instead of binary values, giving more freedom to describe a song and making it possible to see the proximity of the different music clips based on their similarity.

As for the musical features, this paper makes use of a total of 114 features including spectral contrast algorithm, Daubechies wavelets coefficient histogram (DWCH) and collections from both PsySound[9] and Marsyas, two computer frameworks with feature extraction capabilities. In more detail, Spectral contrast represents 12 features and consists of the relative characteristics of each spectral sub band, reflecting the distribution of harmonic components. DWCH represent 28 features and have better ability in representing both local and global information. PsySound extracts 44 features including loudness, level, pitch multiplicity and dissonance based on psychoacoustic models and, finally, Marsyas, extracts a total of 30 features including timbral texture, rhythmic content and pitch content.

In the regressor training, three distinct regression algorithms were tested: Multiple Linear Regression (MLR), Support Vector Regression (SVR) and AdaBoost.RT (BoostR), with SVR

[5] http://www.allmusic.com/
[6] http://www.pandora.com/
[7] http://www.last.fm/
[8] http://www.qloud.com/
[9] http://psysound.wikidot.com/

presenting the best results. For improving results and since not all features are relevant or of sufficient quality, the best ones are selected using a feature selection algorithm (FSA) named RReliefF Principal component analysis (PCA) is also used on the ground truth to reduce correlation between arousal and valence.

Evaluation is made based on a database of 195 popular songs selected from Western, Japanese and Chinese albums, with songs selected by being distributed uniformly in each quadrant of the emotion plane and having one certain dominant emotion.

Although the paper has shown that regressions can be used for mood tracking, the final results of this study were weak, with R2 statistics reaching 58.3% for arousal ad 28.1% for valence. Some of the identified limitations are due to the possible dependence between arousal and valence (in part sorted out with PCA), while identified improvements have to do with better features and lyrics usage, addressing the subjectivity issue and evaluating the regression approach with a large-scale database.

Comparison and Conclusions

For all the articles presented previously, the most relevant features are compared in the following table (Table 2). The scale goes from very bad (--) through moderate (+/-) to excellent (++).

Papers	Precision	Granularity	Diversity	Selection
[Carvalho et al., 2005]	+	-	+/-	-
[Li et al., 2003]	+/-	++	++	-
[Li et al., 2004]	+	+/-	+/-	-
[Feng et al., 2003]	++	-	+/-	-
[Liu et al., 2003]	+/++	-	+/-	++
[Yang et al., 2004]	++	+*	--	-
[Lu et al., 2006]	+/++	-	+/-	++
[Meyers, 2007]	+	+	+	
[Yang et al., 2008]	-		+	++

Table 2: Comparison of methods presented in previous papers

* – Requires annotated lyrics to increase precision. Without them the granularity would be much lower.

This analysis showed a negative relation between granularity of emotions taxonomy and the accuracy of the methods. Papers with a high precision were also the ones where a low granularity was used, with only an average of four emotions used. On the other hand, when a higher number of emotions were chosen the results went down, with much lower values for precision being present. As for the learning and evaluating aspect, they were quite similar between all the publications, requiring a large database of songs with the present emotions manually identified by someone.

The main conclusions taken from these papers are that, first of all is clear that there is no perfect method for mood detection yet and probably there never will be. Nevertheless from an application point of view, there are methods that may be sufficient, depending on the needs for precision and granularity. There is also an urgent need for a database of songs from all genres correctly identified with a set of standardized model of human emotions than can be used to evaluate different algorithms correctly.

As for the subject of this work – mood tracking –, some of the latest papers started to approach this issue, finally looking to moods as something that change during a song. Both [Lu et al., 2006] and [Yang et al., 2008] suggest different approaches to the mood tracking problem. We will follow the later since it fits better in our objectives.

2.3. Mood Tracking

Tracking mood changes is a relatively new subject inside MIR and something that has not been a target of significant research yet. Given this almost embryonic phase, it is therefore normal that few papers tackling this issue have been produced and made available, with [Yang et al., 2008] being the most interesting one. In it, Yang tries to predict mood by using a regression approach. A classification model is first obtained and then small song segments are individually, estimating in this way arousal and valence values through the song. A different strategy is employed in [Lu et al., 2006]. This one is more complex and uses more information, analyzing the songs twice (two passes) to guess possible mood change boundaries and identifying only mood changes between the four clusters.

Below, we describe the three MEVD suited to the Thayer’s mood model proposed in [Yang et al., 1008]

The Fuzzy Approach

This approach uses Thayer’s model of mood and for each input sample a fuzzy vector is assigned, indicating the strength of each emotion class (one of the four quadrants in the model) by fuzzy classifiers. The fuzzy vector $\boldsymbol{\mu}$ for the four emotion classes is expressed as (1):

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \boldsymbol{\mu}_4\}, \sum_{i=1}^{4} \boldsymbol{\mu}_i = \mathbf{1} \qquad (1)$$

where $\boldsymbol{\mu}_i \geq \mathbf{0}$ is the relative strength of class $\boldsymbol{i}$, using the class with highest strength as the classification.

To compute AV values the geometric relationship of the classes are exploited, using the following transformation (2), (3):

$$a = \mu_1 + \mu_2 - \mu_3 - \mu_4, \quad (2)$$

$$v = \mu_1 + \mu_4 - \mu_2 - \mu_3, \quad (3)$$

This approach does have, however, one major flaw. Geometric relationship between arousal and valence is inexact and performing arithmetic operations on AV values is not accurate.

The System Identification Approach (System ID)

The idea behind System ID is to use a system identification technique to model music emotion as a function of 18 musical features. In [Korhonen et al., 2006], the process of audio analysis is performed in intervals of one second to extract audio features from western classical music datasets. Results show average $\boldsymbol{R^2}$ statistics of 78.4% for arousal and 21.9% for valence.

This approach is better than the previous one, not using any geometric operation and showing good results. However there is a crucial need in temporal information, as System ID computes the AV values by using the temporal relationship between segments, making it useless in MER, since this information is not available.

The Regression Approach

The approach proposed in the paper is based on regression training. Regression theory is a well studied and proven theory aiming at predicting a real value from observed variables or features in the past. There are a good number of advantages by using a regressor in this kind of problems: has a sound theoretical foundation, allows easy performance analysis and optimization and usually provides reliable results. It is also important to note that unlike previous strategies, it does not rely on temporal information or on geometric operations.

The problem can be described as, given $\boldsymbol{N}$ inputs $(\boldsymbol{x_{i,}y_i})$, $\boldsymbol{1 \leq i \leq N}$, where $\boldsymbol{x_i}$ is a feature vector for $\boldsymbol{i}$th input sample, and $\boldsymbol{y_i} \in \mathbb{R}$ is the real value to be predicted for the $\boldsymbol{i}$th sample, the regression system trains a regressor $\boldsymbol{R}(\cdot)$ such that the mean squared error $\boldsymbol{\varepsilon}$ is minimized (4):

$$\boldsymbol{\varepsilon} = \frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\left(\boldsymbol{y_i} - \boldsymbol{R}(\boldsymbol{x_i})\right)^{\boldsymbol{2}}, \quad (4)$$

where $\boldsymbol{R}(\boldsymbol{x_i})$ is the prediction result for the $\boldsymbol{i}$th sample.

To use regression theory to predict AV values directly a series of points have to be taken into consideration, being the most relevant:

- The domain of $\mathbb{R}$, confined to [-1, 1], according to Thayer's model.

- Regressor accuracy is dependent on the relevance of the selected audio features.
- Various regression algorithms should be tested in order to select the best one. In Yang et al.'s study, the selected ones were Multiple Linear Regression (MLR), Support Vector Regression (SVR) and AdaBoost.RT (BoostR).
- Two regressors are needed, in order to predict both arousal ($\boldsymbol{R_A}$) and valence ($\boldsymbol{R_V}$).
- The training strategy is important. Since there is a certain dependency between AV, a study is needed to verify if accuracy improves by taking into consideration this dependency when training the regressors (instead of training them independently).

Comparison and Conclusions

A brief comparison between the previously presented mood tracking methods can seen in Table 3.

Name	Accuracy	Temporal Information	Geometric Operation
Fuzzy Approach	N/A	Not needed	Needed
System ID Approach	78.4% (a) 21.9% (v)	Needed	Not Needed
Regression Approach	58.3% (a) 28.1% (v)	Not Needed	Not Needed

Table 3: Comparison on mood tracking methods (adapted from [Yang et al., 2008])

Although there are a few number of different mood tracking methods, they all present some limitations or low accuracy. Given the previous comparison, we will follow the regression approach presented by Yang [Yang et al., 2008]. Even if the results were not the best, it is the one that has more future in MER, not needing to exploit geometric or temporal information that is not always available. Using better audio features, identifying possible limitations and improving or merging new approaches will improve the results.

2.4. Audio Features

Audio features extraction is probably the most important step in music emotion recognition. These "features" are used to represent characteristics present on audio signals and naturally their use is frequent in most, if not all, works in the field. However, the general strategy applied in most approaches focus on developing complex classifiers and implementing good processes of feature selection, classification algorithms and tuning parameters, not always explaining why the initial feature sets were chosen and what they represent or in other cases leaving that to background, picking the ones what were referenced in previous works.

Taking this into account, we discuss in this section the existent features for our problem, explaining what they represent and how they work.

2.4.1. Intensity

Root-Mean-Square Energy (RMS)

Root-Mean-Square is used to measure the power of a signal over a window. The global energy of a signal x can be computed by taking the root average of the square of the amplitude (RMS) [McEnnis et al., 2005], as shown below (5):

$$x_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} \tag{5}$$

Root-Mean-Square Derivative

This feature represents the window-to-window change in RMS. It is an indication of change in signal power.

Root-Mean-Square Variability

The standard deviation of the RMS of the last N windows (jAudio, introduced in section 2.6.1, assigns this parameter a value of 100).

Less-Than-Average Energy

The energy curve can be used to get an assessment of the temporal distribution of energy, in order to see if its remains constant throughout the signal, or if some frames have more contrast than others. One way to estimate this consists in computing the low energy rate, i.e. the percentage of frames showing less-than-average energy [Tzanetakis et al., 2002; cited in Lartillot, 2009], as shown in Figure 7.

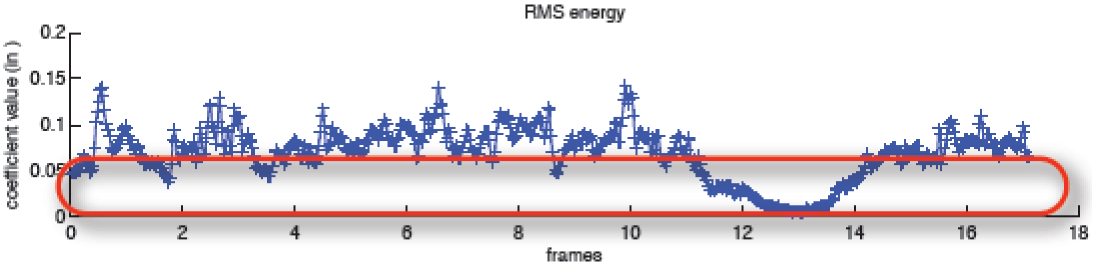


Figure 7: Energy curve with lower-than-average energy frames highlighted [Lartillot, 2009].

Fraction of Low Energy Frames

This feature is defined as the fraction of the 100 previous windows whose RMS is less than the mean RMS. This can indicate how much of a signal section is quiet relative to the rest of the signal section, an indication of the variability of the amplitude of windows.

2.4.2. Rhythm

Rhythmic Fluctuation

This feature represents the rhythmic periodicity along auditory channels. This rhythmic feature is based on spectrogram computation transformed by auditory modeling and then spectrum estimation in each band [Pampalk et al., 2002; cited in Lartillot, 2009]. According to [Lartillot, 2009], this process can be described in two steps:

- First the spectrogram is computed on frames of 23 ms and half overlapping, then the Terhardt outer ear modeling is computed, with Bark-band redistribution of the energy, and estimation of the masking effects, and finally the amplitudes are computed in dB scale.
- Then a FFT is computed on each Bark band, from 0 to 10 Hz. The amplitude modulation coefficients are weighted based on the psychoacoustic model of the fluctuation strength [Fastl, 1982; cited in Lartillot, 2009]. The result is a matrix of the rhythmic periodicities for each different Bark band.

Tempo

Tempo is the speed or pace of a given music piece. In modern music is usually indicated in beats per minute (BPM). Tempo is usually estimated by detecting periodicities from the onset detection curve.

Strength of Strongest Beat

This estimates how strong the strongest beat in the beat histogram is compared to other potential beats [McKay, 2005].

Beat Sum

Represents the sum of all bins in the beat histogram. This is a good measure of the importance of regular beats in a signal [McKay, 2005].

2.4.3. Timbre

Attack Time

Attack time is the estimation of temporal duration for a signal to rise to its peak (e.g., in amplitude), as shown in Figure 8.

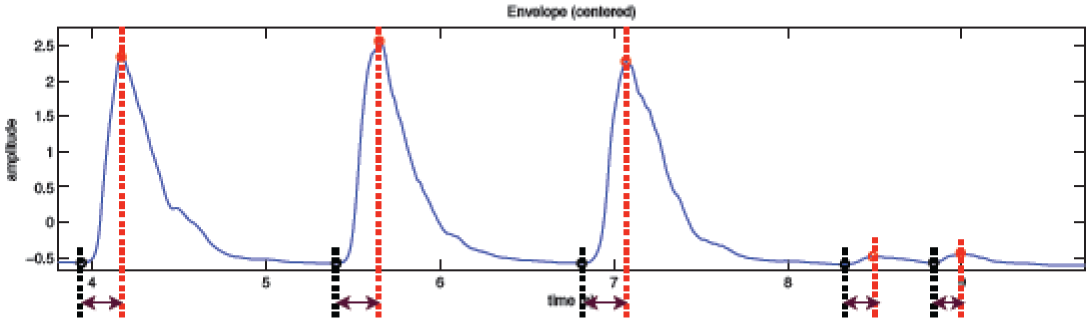


Figure 8: Attack Time detection [Lartillot, 2009]

Attack Slope

Attack slope is another good description of the attack phase. It consists on calculating the average slope of the entire attack phase, since its start to the peak as shown in Figure 9.

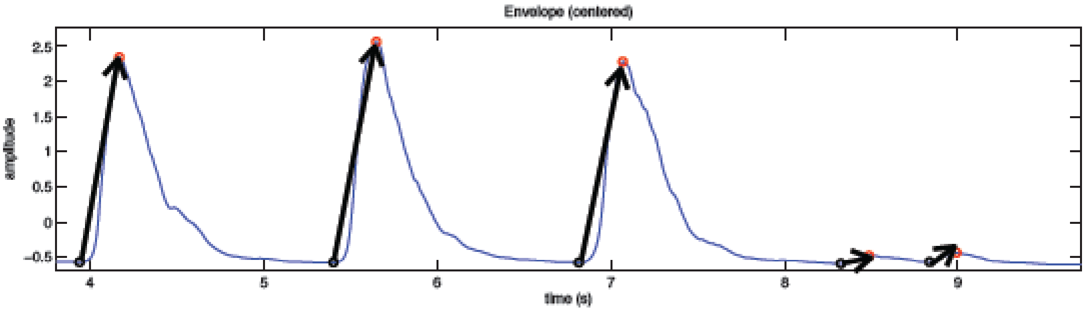


Figure 9: Attack Slope example [Lartillot, 2009]

Zero Crossing Rate

Zero Crossing Rate represents the number of times the waveform changes sign in a window (passes the X-axis – see Figure 10). It can be used as a simple indicator of noisiness. As an example, heavy metal music, due to guitar distortion and heavy percussion, will tend to have much higher zero crossing values than classical music.

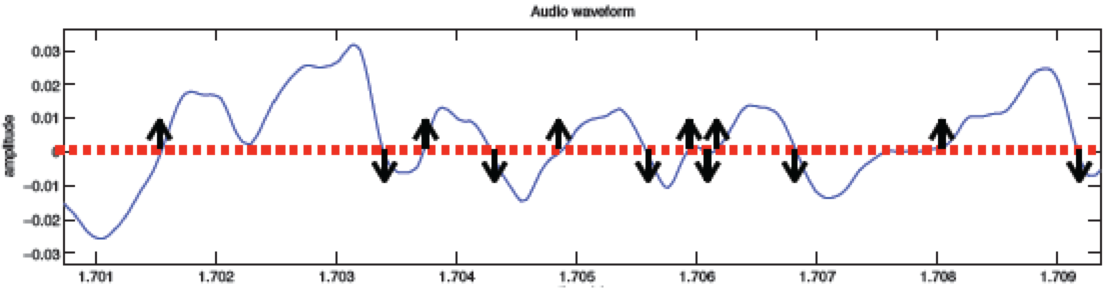


Figure 10: Zero Crossing Rate - Waveform crossing the X-Axis [Lartillot, 2009]

Zero Crossing Rate can be calculated using (6):

$$\boldsymbol{Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - \ sign(x[n-1])|}\,, \tag{6}$$

where the $\boldsymbol{sign}$ function is 1 for positive arguments and 0 for negative arguments and $\boldsymbol{x}[\boldsymbol{n}]$ is the time domain signal for frame *t* [Tzanetakis, 2002].

Zero Cross Derivative

The absolute value of the window-to-window change in Zero Cross, returns an indication of change of frequency as well as noisiness.

Spectral Roll Off

Spectral Roll Off is often used as an indicator of the skew of the frequencies present in a window. It consists in finding the fraction of the total energy (Hz) that is contained below a given percentage, as shown in Figure 11. The percentage varies among authors, with 85% being the current default value for most frameworks following [Tzanetakis et al., 2002; cited in Lartillot, 2009], while [Pohle et al., 2005; cited in Lartlillot, 2009] propose 95%.

According to [Tzanetakis, 2002], mathematically the Spectral Roll Off is defined as the frequency R_t below which 85% of the magnitude distribution is concentrated (7):

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^{N} M_t[n]\,, \tag{7}$$

where $M_t[n]$ is the magnitude of the Fourier transform at frame *t* and frequency bin n.

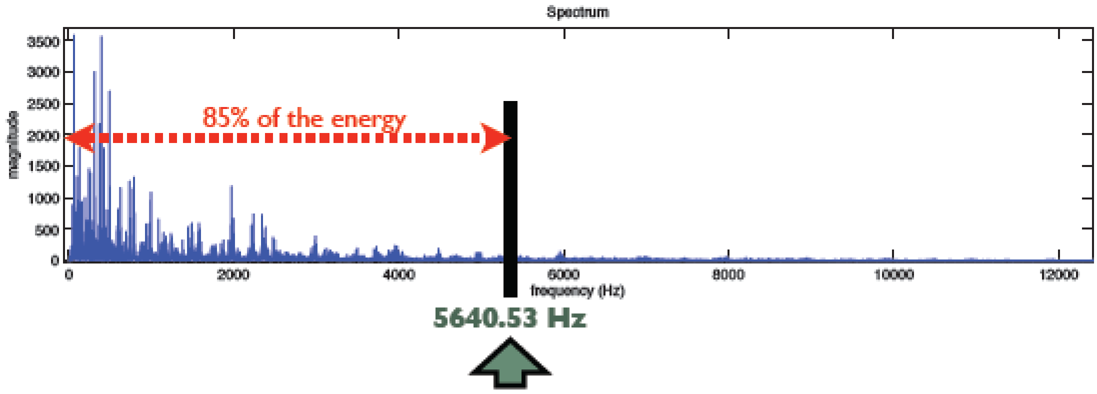


Figure 11: Spectral Roll Off with frequency [Lartillot, 2009]

High Frequency Energy

High frequency energy, also called brightness by some authors, consists in fixing a minimum frequency value, and measuring the amount of energy above that frequency, as exemplified on Figure 12. The result is expressed as a number between 0 and 1, with the proposed cut-off frequency values varying between 1500 Hz [Lartillot, 2009], 1000 Hz [Laukka et al., 2005; cited in Lartillot, 2009] and 3000 Hz [Justin, 2000; cited in Lartillot, 2009].

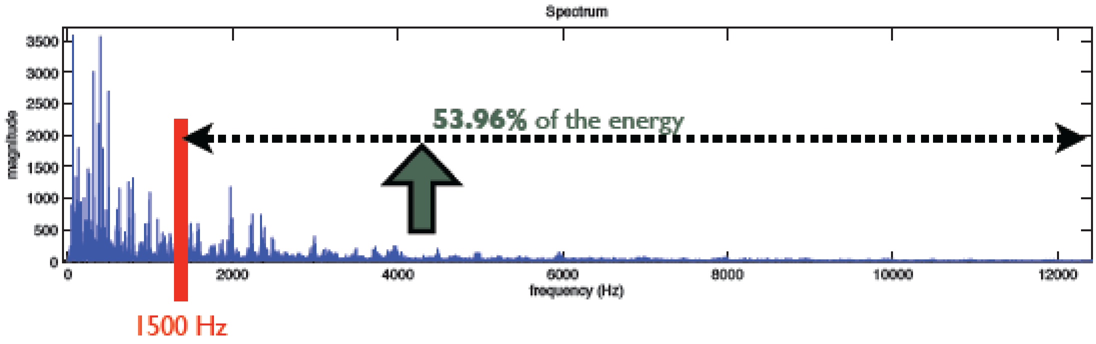


Figure 12: Spectral Roll Off using % [Lartillot, 2009]

Spectral Flux

Spectral Flux is a measure of the amount of spectral change in a signal, i.e., the distance between adjacent frames. Spectral flux has also been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre [Gray, 1975; cited in Tzanetakis, 2002].

According to [Tzanetakis, 2002] the spectral flux is defined as the squared difference between the normalized magnitudes of successive spectral distributions (8):

$$\boldsymbol{F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2}, \tag{8}$$

There, $\boldsymbol{N_t[n]}$, $\boldsymbol{N_{t-1}[n]}$ are the normalized magnitude of the Fourier transform at the current frame $\boldsymbol{t}$, and the previous frame $\boldsymbol{t-1}$ respectively.

Mel-Frequency Cepstral Coefficients (MFCC)

MFCCs offer a description of the spectral shape of the sound. The frequency bands are positioned logarithmically (on the Mel scale), which approximates the human auditory system's response more closely than the linearly-spaced frequency bands.

The calculation of this feature works as follows [Tzanetakis, 2002]. After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Then, in order to decorrelate the resulting feature vectors, a Discrete Cosine Transform is performed. Usually, only the first 13 components are returned since most of the signal information tends to be concentrated in a few low-frequency components of the discrete cosine transform (DCT). These 13 coefficients are mostly used for speech representation but [Tzanetakis, 2002] states that the first five coefficients are adequate for music representation.

An image detailing how the process occurs using the MIR ToolBox is shown in Figure 13.

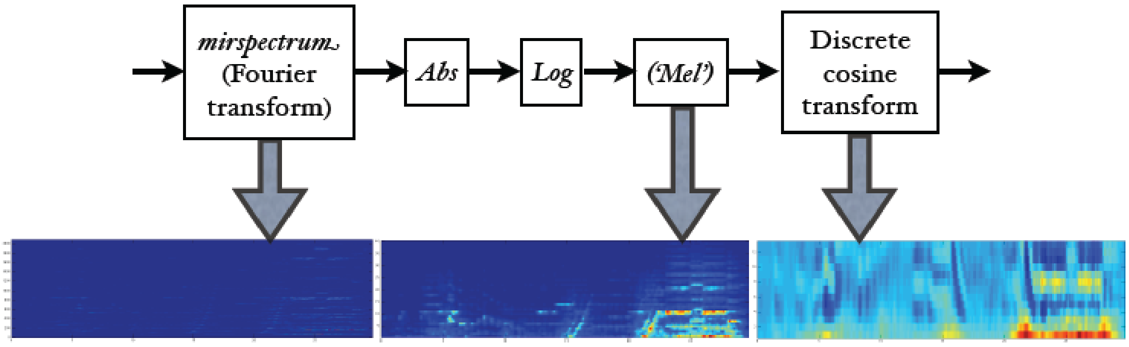


Figure 13: MFCC implementation according to [Lartillot, 2009]

Sensory Dissonance (Roughness)

Sensory dissonance, also known as Roughness, is related to the beating phenomenon that occurs whenever a pair of sinusoids is close in frequency. [Plomp, 1965; cited in Lartillot, 2009] propose an estimation of sensory dissonance depending on the frequency ratio of each pair of sinusoids, as represented in Figure 14.

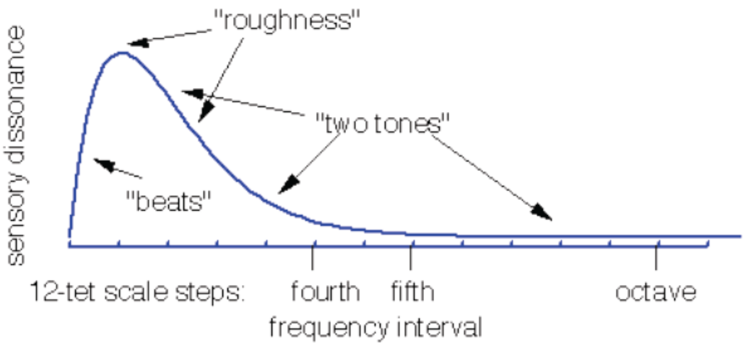


Figure 14: Sensory Dissonance depending on frequency ratio [Lartillot, 2009]

Spectral Peaks Variability (Irregularity)

Spectral peaks variability or irregularity is, as the name indicates, the degree of variation of the successive peaks of the spectrum.

According to [Lartillot, 2009], the MIR ToolBox has two distinct approaches to Spectral Peaks Variability calculation:

- The default approach is based on [Jensen, 1999; cited in Lartillot, 2009], where the irregularity is the sum of the square of the difference in amplitude between adjoining partials (9):

$$\left(\sum_{k=1}^{N}(a_k - a_{k+1})^2\right) / \sum_{k=1}^{N} a_k^2 \qquad (9)$$

- The second approach is based on [Krimphoff et al., 1994; cited in Lartillot, 2009], where the irregularity is the sum of the amplitude minus the mean of the preceding, current and next amplitude (10):

$$\sum_{k=2}^{N-1}\left|a_k - \frac{a_{k-1} + a_k + a_{k+1}}{3}\right| \qquad (10)$$

Spectral Centroid

The Spectral Centroid is a measure of spectral shape. Higher centroid values correspond to "brighter" textures with more high frequencies. It is defined as the center of gravity of the magnitude spectrum of the STFT (11):

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] * n}{\sum_{n=1}^{N} M_t[n]}, \qquad (11)$$

where $M_t[n]$ represents the magnitude of the Fourier transform at frame t and frequency bin n [Tzanetakis, 2002].

According to [Gray, 1975; cited in Tzanetakis, 2002], spectral centroid has been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre.

Linear Prediction Reflection Coefficients

Linear prediction reflection coefficients (LPRC) are used in speech research as an estimate of the speech vocal tract filter [Makhoul, 1975; cited in Tzanetakis, 2002]. They are also used in musical signals.

Linear Spectral Pairs

Linear Spectral Pairs (LSP) or line spectral frequencies (LSF) are used to represent linear prediction coefficients (LPC) for transmission over a channel. Because LSPs are not overly sensitive to quantization noise and stability is easily ensured, LSP are widely used for quantizing LPC filters. For this reason, LSPs are very useful in speech coding.

Strongest Frequency via Spectral Centroid

This feature is an estimate of the strongest frequency component of a signal, in Hz, found via the spectral centroid [McKay, 2005].

Spectral Crest Factor

The Spectral Crest Factor (SCF) or peak-to-average ratio (PAR) is a measurement of a waveform, calculated from the peak amplitude of the waveform divided by the RMS value of the waveform (12).

$$\boldsymbol{C} = \frac{|\boldsymbol{x}|_{\boldsymbol{peak}}}{\boldsymbol{x}_{\boldsymbol{rms}}}, \tag{12}$$

Spectral Flatness Measure

Spectral Flatness Measure (SFM) is a measure used to characterize an audio spectrum. A high SFM indicates that the spectrum has a similar amount of power in all spectral bands – this would sound similar to white noise, and the graph of the spectrum would appear relatively flat and smooth. A low spectral flatness indicates that the spectral power is concentrated in a relatively small number of bands – this would typically sound like a mixture of sine waves, and the spectrum would appear "spiky".

The SFM is calculated by dividing the geometric mean of the power spectrum by the arithmetic mean of the power spectrum (13)

$$\boldsymbol{SFM} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} \boldsymbol{x(n)}}}{\frac{\sum_{n=0}^{N-1} \boldsymbol{x(n)}}{\boldsymbol{N}}}, \tag{13}$$

where $x(n)$ represents the magnitude of bin number n.

Inharmonicity

Inharmonicity measures the amount of partials that are not multiples of a given fundamental frequency, $\boldsymbol{f_0}$, as shown on Figure 15. Inharmonicity influences the timbric perception of a given sound.

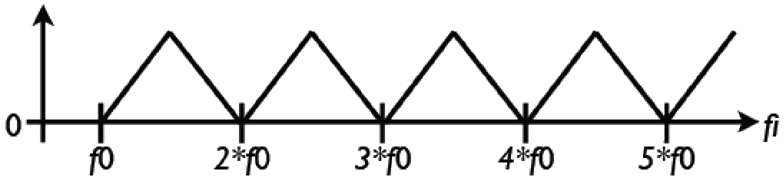


Figure 15: Fundamental frequency (f0) and its multiples [Lartillot, 2009]

2.4.4. Pitch

Pitch (f0)

Pitch represents the perceived fundamental frequency of a sound. It is one of the three major auditory attributes of sounds, along with loudness and timbre. Pitch (as an audio feature) typically refers to the fundamental frequency of a monophonic sound signal and can be calculated using various different techniques [Tzanetakis, 2002]. One of the methods employed in Marsyas to calculate pitch is the YIN algorithm [Cheveigné et al., 2002]

Strongest Frequency via FFT maximum

This is an estimate of the strongest frequency component of a signal, in Hz, found via the FFT bin with the highest power [McKay, 2005].

2.4.5. Tonality

Key (Tonal Center Positions)

This feature gives a broad estimation of tonal center positions and their respective clarity [McKay, 2005].

Mode

The mode is an important feature that estimates the difference between major and minor keys (the modality of a piece).

Tonal Centroid

Tonal centroid is a 6-dimensional feature vector. It corresponds to a projection of the chords along circles of fifths, of minor thirds, and of major thirds [Harte et al., 2006; cited in Lee, 2007]. It is based on the Harmonic Network or Tonnetz, which is a planar representation of pitch relations where pitch classes having close harmonic relations such as fifths, major/minor thirds have smaller Euclidean distances on the plane. By calculating the Euclidean distance between successive analysis frames of tonal centroid vectors, they successfully detect harmonic changes such as chord boundaries from musical audio (exemplified in Figure 16) [Lee, 2007].

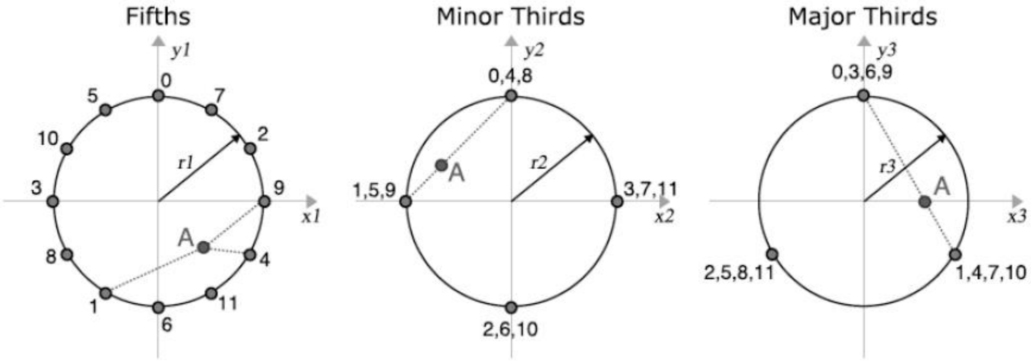


Figure 16: Tonal Centroid for A major triad (pitch class 9, 1 and 4) is shown at point A [Lee, 2007]

Harmonic Change Detection Function

The Harmonic Change Detection Function (HCDF) is the flux of the tonal centroid [Harte et al., 2006; cited in Lartillot 2009]

2.4.6. Musical Features

Musical features are a set of small features introduced in [Tzanetakis, 2002], which are based on musical content and information extracted by previously discussed features. They can belong to two distinct categories: rhythmic and pitch content, and will be usually referred as Musical Features only. Details of these features are presented below:

Rhythmic content features, calculated with recourse to the Beat Histograms (BH) of a song:

- A0, A1: relative amplitude (divided by the sum of amplitudes) of the first (A0), and second (A1) histogram peak
- RA: ratio of the amplitude of the second peak divided by the amplitude of the first peak
- P1, P2: Period of the first (P1) and second (P2) peak in BPM
- SUM: overall sum of the histogram (indication of beat strength)

Pitch content features, calculated with recourse to Folded and Unfolded Pitch Histograms (FPH and UPH):

- FA0: Amplitude of the maximum peak of the folded histogram. This corresponds to the most dominant pitch class of the song. For tonal music, this peak will typically correspond to the tonic or dominant chord. This peak will be higher for songs that do not have many harmonic changes.
- UP0: Period of the maximum peak of the unfolded histogram. This corresponds to the octave range of the dominant musical pitch of the song.
- FP0: Period of the maximum peak of the folded histogram. This corresponds to the main pitch class of the song.
- IPO1: Pitch interval between the two most prominent peaks of the folded histogram. This corresponds to the main tonal interval relation. For pieces with simple harmonic structure, this feature will have a value of 1 or -1, corresponding to fifth or fourth intervals (tonic-dominant).
- SUM: The overall sum of the histogram. This feature is a measure of the strength of the pitch detection.

2.4.7. Statistical Features

Statistical features are a class of features calculated using some of the previously defined features. It is possible to extract statistical data from almost all features, typically 1st and 2nd order statistics like means and standard deviations, as well as higher-order statistics (skewness, kurtosis and others).

2.4.8. Feature Relevance

One of the main goals of this work is not only to gain knowledge on the existent audio features that may be applied to mood detection but also to understand the relevance of each one on this topic. A similar study was previously done by Yang and the results are presented at [Yang et al., 2008]. Those results served as a base to our research, giving us an idea of what features were indeed more relevant for mood prediction for both arousal and valence. Nonetheless, tests were conducted to assess the relative importance of different features.

To conduct these tests, Forward Feature Selection (FFS) was employed. It works generally as follows. Starting from an empty feature set, a model consisting of single feature (each feature in the set is iteratively experimented) is first obtained. Then, the feature with best performance is added to the feature set. This procedure is repeated until all features are added. When finished, the result is a ranking of the feature relevance and the "optimal" feature sub-set to use.

2.5. Classification Methods: Support Vector Machines

Statistical Classifiers are supervised machine learning procedures used to evaluate individual items based on their characteristics and correctly placing them into groups, using as a base a training set of previously labeled items.

Although there are innumerous classification methods available, we will restrict this section to the one providing the best accuracy, as reported in the literature, e.g., [Yang et al., 2008]: Support Vector Machines.

Support vector machines (SVMs) is one of the most used classifiers today, composed by a set of related supervised learning methods used for classification and regression.

SVMs work by analyzing a given a set of training examples, previously identified by belonging to a given category. An SVM training algorithm builds a model that predicts in which category a new example falls into. This functioning can be explained as points in space mapped so that the examples from the training set of different categories are divided by a clear gap that is as wide as possible, as demonstrated on Figure 17. This separation between classes does not need to be linear. The new, unclassified, examples are then mapped into that same space and predicted to be of a determined category based on which region of the gap they fall on.

There are two main categories for support vector machines: Support Vector Classification (SVC), used with classes or categories as exemplified in the picture below, and Support Vector Regression (SVR), which are used for linear regression, to train and predict data using real values as annotations (for example, arousal and valence values instead of classes representing the four quadrants for example).

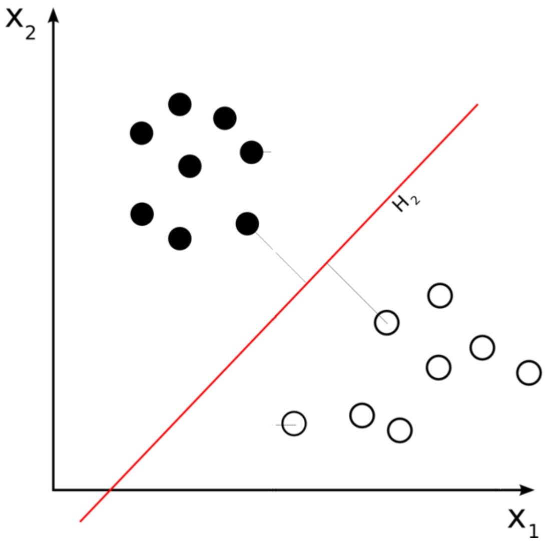


Figure 17: SVM – Examples separated by the maximum margin (H2)

2.6. Existing Frameworks / Platforms

Even though MIR and MER are relatively new areas of research, some work has already been done related to feature extraction and audio analysis tools, providing us with frameworks able to simplify audio processing tasks and allowing for more advanced tools to be created based on them. Taking this into consideration, we will approach here three of the most important frameworks to the date, the ones considered to be of highest interest for this work. Still, the reader should note that there are a lot more than these 3, with different ambits and goals, so names like Psysound[10], CLAM, MK2 and others should also be taken into consideration in future researches.

2.6.1. JAudio

jAudio[11] is a software package started in 2005 at McGill University for extracting low and high level features from audio files as well as for iteratively developing and sharing new features. The initial idea was to provide a framework to eliminate the effort in calculating from the signals, providing by default a wide range of analysis algorithms suitable to MIR tasks. To complement this and decrease the slope of the learning curve, the application also provides an easy to use graphic user interface (GUI), making feature selection and audio processing straightforward. Batch processing and other features are also provided via a command line interface, making it possible to create desired user scripts. The output is obtained done in Weka's ARFF format or the ACE XML file, to be later processed with machine learning

[10] http://psysound.wikidot.com/
[11] http://jmir.sourceforge.net/jAudio.html

framework such as ACE. jAudio is an open source project currently developed using the Java language, which makes it slower and heavier than some of the alternatives but also portable. Some of the positive aspects are the high number of audio features available and the portability.

2.6.2. Marsyas

Marsyas[12], or Music Analysis, Retrieval and Synthesis for Audio Signals, is a software framework developed for audio processing with specific emphasis on Music Information Retrieval applications. It has been designed and written by George Tzanetakis with the collaboration of students and researchers from around the world. It was a pioneer in the area, being one of the first frameworks in MIR. Marsyas has been used for a variety of projects in both academia and industry, and it is known to be computationally efficient, due in part to the fact of being written in highly optimized C++ code (MIREX 2008 results[13]). It is an open source framework, accepting contributions for all developers willing to help, and is capable of outputting results to the Weka's ARFF format. The native integration with Qt[14] makes it also possible to create full applications with GUI.

Some of the least polished aspects are the lower number of audio features when comparing to the alternatives and its weak documentation. Another verified problem is its complicated interface and syntax to build and control the audio processing networks.

2.6.3. MIR ToolBox

The MIR toolbox[15] is an integrated set of functions written in Matlab, that are specific to the extraction of musical features such as pitch, timbre, tonality and others. The project is designed in a modular way, where the different algorithms can be decomposed into smaller, elementary functions and mechanisms. This approach allows users to interact with these minimal blocks and combine them in different and original ways to generate new features.

Batch processing of several audio files or even folders is possible. The great number of low and high-level features combined with the adaptive syntax to create new functions and also the existing output methods, allowing not only to export information but also to have graphic visualization, makes this a great framework. The provided documentation is good and also something to take into consideration, especially when compared with the other frameworks. On the less bright side of the framework are its dependencies, among which

[12] http://marsyas.sness.net/
[13] http://www.music-ir.org/mirex/2008/index.php/Audio_Music_Mood_Classification_Results
[14] http://qt.nokia.com/. Qt is an excellent cross-platform application and UI framework, with an LPGL licensed version, enabling the creation of portable applications across Windows, Mac, Linux/X11, embedded Linux, Windows CE and Symbian without rewriting the source code.
[15] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

stands out obviously MathWorks' Matlab[16] and MathWorks' Signal Processing Toolbox[17], due to the fact of being commercial products.

2.6.4. Frameworks Comparison

Following, a small comparison of the frameworks will be presented, with emphasis on the available features (Table 4), classifiers (Table 5) and also the most relevant technical aspects of the project (Table 6).

Implemented Features

Feature	Feature class	jAudio	Marsyas	MIR toolbox	Others
Fraction Of Low Energy Frames	intensity	x			
Less-Than-Average Energy	intensity	x		x	
Root-Mean-Square Derivative	intensity	x			
Root-Mean-Square Energy	intensity	x		x	
Root-Mean-Square Variability	intensity	x			
Pitch (F0)	pitch		x	x	x
Strongest Frequency Via FFT Maximum	pitch	x			
Beat Sum	rhythm	x	x		
Rhythmic Fluctuation	rhythm			x	x
Strength Of Strongest Beat	rhythm	x		x	
Tempo	rhythm	x	x	x	x
Attack Slope	timbre			x	
Attack Time	timbre			x	
High Frequency Energy (Brightness)	timbre			x	x
Inharmonicity	timbre			x	
Linear Prediction Reflection Coefficients	timbre	x	*		
Linear Spectral Pairs	timbre		x		
Mel-Frequency Cepstral Coefficients	timbre	x	x	x	x
Sensory Dissonance	timbre			x	x
Spectral Centroid	timbre	x	x	x	x
Spectral Crest Factor	timbre		x		
Spectral Flatness Measure	timbre		x		
Spectral Flux	timbre	x	x	x	x
Spectral Peaks Variability (Irregularity)	timbre	x		x	x
Spectral Roll Off	timbre	x	x	x	x

[16] http://www.mathworks.com/products/matlab/
[17] http://www.mathworks.com/products/signal/

Strongest Frequency Via Spectral Centroid	timbre	x			
Zero Cross Rate	timbre	x	x	x	x
Zero Cross Derivative	timbre	x			
Harmonic Change Detection Function	tonality			x	
Key (Tonal Center Positions)	tonality			x	x
Modality	tonality			x	x
Tonal Centroid	tonality			x	
Musical Content Features	-		x		

Table 4: Available features in each framework

(* LPRC is obtained from Linear Prediction Cepstral Coefficients (LPCC), which is supported by the framework.)

Implemented Classifiers

Classifiers	jAudio	Marsyas	MIR toolbox
K-Nearest-Neighbour		x	x
Gaussian Mixture Model		x	x
Support Vector Machines		x*	

Table 5: Classifiers available in each framework

(* Marsyas does not support SVRs)

Technical Aspects

	Features	Classifiers	Interface	Performance	Language	Documentation
jAudio	~30 / 139	0	GUI	low	Java	Unknown
Marsyas	~30	3	CLI, source code	high	C++	Weak
MIR toolbox	~40	3	Adaptative syntax	low	Matlab	Good

Table 6: General details of frameworks studied

Although having the lowest number of implemented features by default (something that can be changed by implementing needed ones) and lacking a good documentation that would be helpful in the initial phase, Marsyas was the framework used due to its much higher computational performance (MIREX 2008), to the fact that it was written in C++, integrated with Qt and independent of other commercial software. jAudio has the highest number of features, however many of them are statistical features, like means and standard deviations, obtained from other features.

2.7. Test Collections and Evaluation Procedures

Evaluation of results and system performance is essential to verify possible errors, to tune settings and especially to measure the application accuracy. To this end, a test collection is needed. The primary aspects that make a good and useful test collection for this project are:

- Number of music pieces – a low number of songs can compromise the training and testing results.
- Music genres available – it is better to have various genres, measuring how the system performs globally. Usually few genres or just one can lead to the creation of systems that are too specific or just perform well inside that genre.
- Geographical location – We have a global world with different musical tastes between regions and emotions are also something very subjective, not only from person to person but also between cultures. A database featuring songs from different cultures and not only western could be valuable to study how the system performs.
- Annotation methods – The methods used to annotate songs are important, if possible made by more than one person, since having a good, reliable collection is fundamental to the entire testing phase.
- Distribution of songs – it is essential to have a uniform number of songs in each category. In our case, to have a similar number of pieces in each of the four Thayer's quadrants.

Taking the previous points into consideration, we decided to use a data collection and annotations kindly provided by Yi-Hsuan Yang, which was previously used in [Yang et al., 2008]. The dataset consists of 194 clips of 25 seconds each (in WAV PCM format, 22050 Hz sampling rate, 16 bits quantization, mono), most being of Pop genre songs selected from Western, Chinese and Japanese albums. The authors describes the dataset as being uniformly distributed (48/49 in each of the four quadrants), with each clip expressing a clear dominant mood. Clips of 25 seconds only are used instead of entire songs due to the fact that mood changes tend to occur during entire songs. These 25 seconds were manually trimmed to better represent the music and its dominant emotion. Later, complete songs may be used to better test mood tracking accuracy.

The annotation process was complex and involved 253 volunteers with several backgrounds (from regular people to philosophy and music experts), analyzing ten clips each. The subjects were asked to listen and label ten random clips from the collection, giving values between -1.0 and 1.0 to arousal and valence, according to the emotion evoked by the music clip. Not only melody but also lyrics and singing (vocal) of the song were taken into consideration by subjects, who were given the opportunity to hear the clips more than one time.

The consistency of the ground truth was evaluated by averaging subjects' annotations. For each song, a larger standard deviation means a less accurate annotation, due to subjectivity or mood ambiguity in the song. These calculations for arousal / valence values resulted in a deviation of about 0.3, with 95% confidence interval of ±0.2, which is an acceptable value and shows the subjectivity issues that exist when dealing with emotions. In addition to this test, a second one was conducted that consisted in repeating the process with 22 of the same subjects, retesting the same songs two months later in order to guarantee the reliability of the results. The large the difference was, the less repeatable the subjective test could be. The results showed that more than half of the annotations varied only in 0.1, confirming that the test is repeatable.

Concluding, the annotations are reliable, nevertheless showing a small degree of inconsistence on the ground truth, something that is reasonable due to the subjectivity existent in music perception and emotions in nature.

However, after annotating the songs in the 4 quadrants with specific AV values, it turned out that some clips moved to different quadrants. For example, songs originally labeled as belonging to the second quadrant had positive valence values, moving them to the first quadrant. As a result, the initial requirement for a balanced dataset was violated, resulting the distribution in Table 7. This uneven distribution has clear impact in the creation of the mood classification model. Unfortunately, this issue was not mentioned in [Yang et al., 2008]. In addition, some songs were annotated with zero arousal or valence. Hence, the songs between two quadrants (having a value of 0 for arousal or valence) were attributed to the quadrant they were more probable to be in according to the verified annotations (e.g. a song between quadrants 1 and 2 were attributed to 1). In Table 7, “Other” represents songs that are exactly placed between two quadrants, by having a value of zero for arousal or valence. “Match” represents the number of songs placed by Yang that were indeed correct.

Quadrants	Yang	Annotation	Annot (%)	Match	Match (%)
1	48	54	27,84%	36	75,00%
2	48	22	11,34%	17	35,42%
3	49	51	26,29%	16	32,65%
4	49	49	25,26%	9	18,37%
Other	0	18	9,28%	-	-
Total	194	194	100,00%	78	40,21%

Table 7: Songs per quadrant: Yang vs. real annotations

Since no annotations for mood changes were available, they had to be created in order to evaluate mood tracking results. To create the dataset, 57 full songs of the 194 titles presented in Yang collection were obtained, leaving out the oriental songs that were impossible to gather. Annotating manually the mood changes over songs is a difficult and time consuming task. Taking that into consideration, we opted to use manual annotations made by two volunteers (students at our department, with music experience), where they marked the quadrant changes over a song as exemplified in Figure 18. This process was done for the entire dataset and the matching rate between each pair was calculated, by counting the time both identified the same quadrant. For testing purposes only the ones with a matching ratio of 80% or higher, 29 songs, were used.

Clearly, the current annotations are not sufficient. To overcome the problem, the number of songs of the dataset will be increased and more volunteers will be used to manually annotate mood changes.

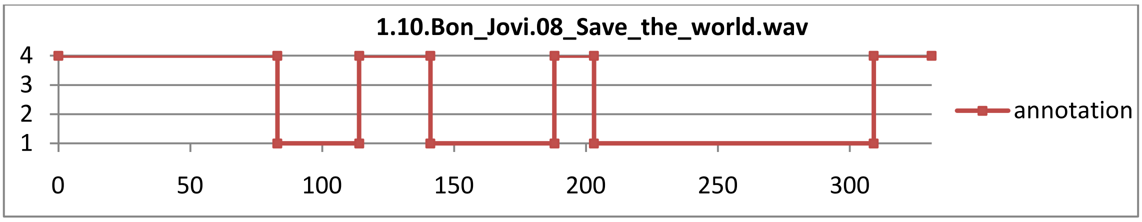


Figure 18: Tracking annotation example

2.8. Followed Approach

After a long research in the area, we finally had sufficient knowledge to choose and detail an approach to be followed in the scope of this project – mood tracking. The main goal of the project was not to innovate in terms of features and algorithms but to create a robust base of work that can be used later for future innovation. Hence, an approach similar to the one followed in [Yang et al., 2008] was followed.

Based on that work, Thayer's model of mood was used, having a continuous view of emotions instead of categorical, plus the simple arousal/valence axis system gives it an advantage comparing to others.

Feature extraction was carried out using the Marsyas framework. Although complex and sometimes unstable, it showed to be fast and with a good variety of features.

For the classification part an extra library was added – libSVM[18]. This C library provides support for SVMs and SVRs and proved to be very powerful, bridging the gaps of Marsyas in this aspect (Marsyas does not support SVR, as mentioned previously).

Several tests were conducted with the developed tool to measure important aspects as speed, computer needs, accuracy and feature relevance. Some limitations were also identified as well as suggestions and mechanisms to surpass them.

The developed application is composed of 3 distinct parts: server, client and backoffice. Currently, a functional prototype of both server and client exists (alpha version), letting the user visualize a song waveform, highlighting mood changes across the four

[18] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

quadrants using different colors. It will also show the distribution of all songs on the DB, using an AV plan as well as selected song details. Other features will be implemented in the near future by me and João, related with playlist creation by either selecting a point or path on the AV graph.

3. Implementation

In this chapter I will present the main decisions about the implementation of the audio analysis core for feature extraction, classification and tracking, as well as the server and client components of the mood detection application. More details about the software engineering process can be found on the appropriated appendix.

3.1. Mood Tracking

3.1.1. Feature Extraction

Marsyas was used to extract features from the audio files. It is powerful and fast and at the same time one of the few free C++ alternatives. The overall Marsyas architecture is described in Appendix C.

The feature extraction process is based on source code of the Marsyas examples, using two different MarSystems[19] networks to extract features depending on the desired result. There are innumerous MarSystems available, each one with a different function.

The most relevant composite MarSystems are:

- Series – the most basic structure for connecting MarSystems (in Series, as the name implies) into dataflow networks.
- Parallel – receives an input with multiple observations (i.e. channels) and sends each observation to a different MarSystem where calculations run in parallel.
- Fanout – similar to Parallel, but takes a single observation and sends a copy of this observation to all the MarSystems inside of it.
- Accumulator – accumulates results of multiple tick[20] process calls to the internal MarSystem. If its *nTimes* control is set to 10, for each tick received by Accumulator, 10 ticks will be sent to its internal MarSystems.

To obtain a unique value for each feature that represents the entire song, normally called a single vector of features, the network described on Figure 19 is used. For mood tracking and similar usages there the need of feature values in small time intervals, usually a

[19] In Marsyas terminology the processing nodes of the dataflow network are called MarSystems and provide the basic building blocks out of which more complicated systems are built. Essentially any audio processing can be expressed as a large composite MarSystem which is assembled by appropriately connected basic MarSystems.

[20] A tick represents an instant of time. Each time the tick() function is called a data slice is propagated across the entire dataflow network.

group of N windows. This is achieved with the network in Figure 20. An extra independent network is also used when extracting a single vector to calculate beat histogram features, giving 18 additional results.

Analyzing the two networks, the main difference that permits a different behavior is the usage of an *Accumulator* MarSystem in the single vector approach, which encloses the *Series* network responsible for feature extraction and processing. This MarSystem accumulates results of multiple tick process calls to its internal MarSystems, generating output only once when all the results are accumulated, as described in [Percival et al., 2009].

As for the features supported, we tried to use all features available in Marsyas. The list is composed by the following timbral features, as well as pitch and beat histogram features (currently only for the single vector approach): Beat Histogram Features, Centroid, Chroma, Flux, LPC derived Cepstral coefficients (LPCC), Linear Spectral Pairs (LSP), Mel-frequency Cepstral Coefficients (MFCC), Pitch Histogram Features, Rolloff, Spectral Crest Factor (SCF), Spectral Flatness Measure (SFM), Tempo and Zero Crossing Rate. These features represent a total of 218 values for tracking and 454 for the single value due to the extra standard deviation and mean calculated by the song statistics *Fanout* MarSystem (218 x 2), plus the extra beat histogram features composed by 18 values.

MarSystem Networks used for feature extraction

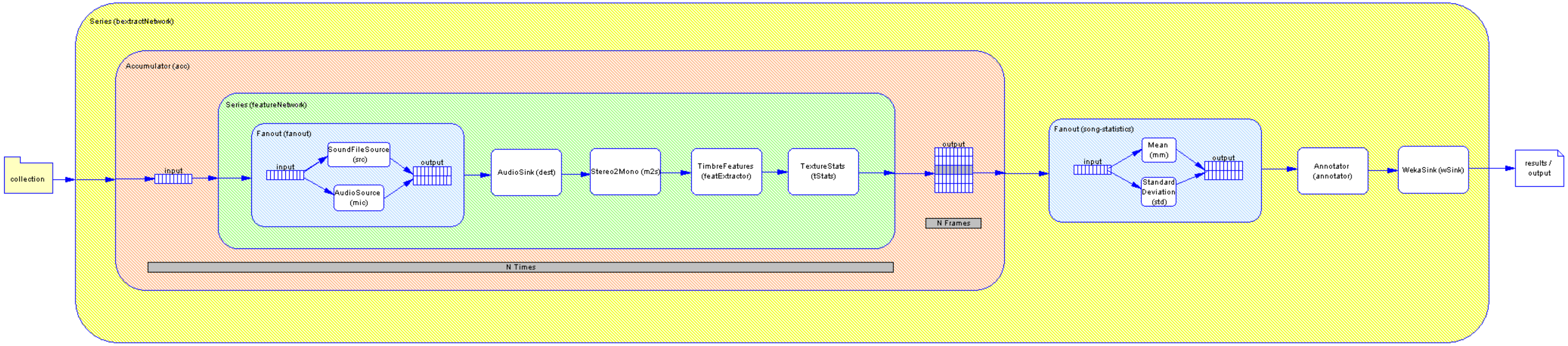


Figure 19: Feature extraction network (single vector)

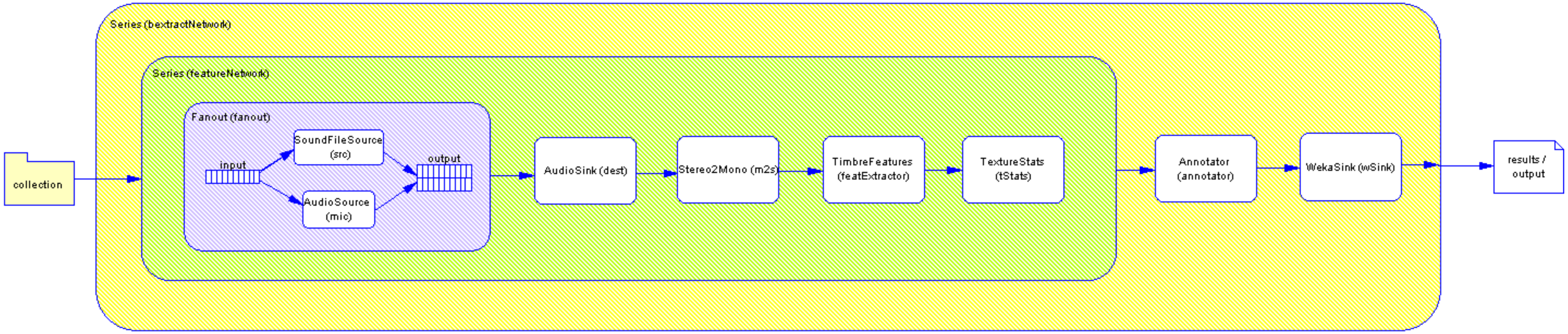


Figure 20: Feature extraction network (continuous/tracking)

Data Normalization

Before being used in other tasks, the feature values need to suffer one last transformation. This transformation consists in scaling all values to be of similar magnitude. Skipping this step would lead to numerical problems resulting from different feature ranges, which would seriously compromise the training and testing results.

To obtain the normalized result, $\boldsymbol{N_t}$, of an original feature value, $\boldsymbol{F_t}$, we apply equation (14). The minimum ($\boldsymbol{min_t}$) and maximum ($\boldsymbol{max_t}$) values for each feature are stored during the extraction.

$$\boldsymbol{N_t} = \boldsymbol{l} + (\boldsymbol{u} - \boldsymbol{l}) \times \frac{\boldsymbol{F_t} - \boldsymbol{min_t}}{\boldsymbol{max_t} - \boldsymbol{min_t}}, \tag{14}$$

$\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower and upper bounds between which the features will be scaled. On our tests we normalized features to the [0, 1] interval ($\boldsymbol{l}$ = 0 and $\boldsymbol{u}$ = 1).

3.1.2. Classification

AV Classification

In the classification process a different library – libSVM – was used. The reason behind this choice was the lack of support for regressions and real number annotations in Marsyas. Although the framework uses the same library to classify when using the SVM MarSystem, it only supports SVC and treats annotations as classes. Adding support for SVR would not be straightforward and as a consequence we decided to use the original library itself, which was easier to use and gave us much more freedom.

To train the SVM model we used the Yang dataset composed of 194 audio files annotated with arousal and valence values. The tests were conducted running 20 repetitions of a 10-fold cross validation, where the songs were divided into ten subgroups, using 9 groups to train and one group to test. This process repeated ten times, guaranteeing that all groups (and consequently all songs) were used for testing and for training.

The predicted results for each song arousal and valence were averaged and used to calculate several statistics:

Sum-Square Error (SSE) – measuring the total deviation of the predicted values from the original annotations (15).

$$\boldsymbol{SSE} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\boldsymbol{2}}, \tag{15}$$

where $\boldsymbol{y_i}$ is the annotation and $\boldsymbol{\hat{y}_i}$ the predicted value.

Total Sum of Squares (SST) – measuring the deviation of the each annotation to the mean value of the annotations (16).

$$\boldsymbol{SST = \sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{16}$$

where $\boldsymbol{y_i}$ is the annotation and $\boldsymbol{\bar{y}}$ the average of all annotation values.

Root Mean Square Error (RMSE) – an estimate of the standard deviation of the predicted values (17).

$$\boldsymbol{RMSE = \sqrt{\frac{SSE}{N}}}, \tag{17}$$

R2 statistics – used to measure how successful the trained model is and how it fits our training and test data (18), with results near 1 meaning the model fits the data perfectly.

$$\boldsymbol{R2 = 1 - \frac{SSE}{SST}}, \tag{18}$$

Overall AV classification accuracy is provided by the R2 statistics, for the sake of comparison with Yang et al.’s results.

Quadrant Classification

As our mood tracking annotations only offer quadrant information (AV pairs were not acquired in the subjective annotation task - see Section 2.7), a model classification into the 4 Thayer’s quadrants must be obtained. Thus, to create the SVM model, instead of using arousal and valence values directly, we transformed these values to quadrants, using them as classes to train a SVC instead of an SVR.

Using this strategy, the entire 194 song clips were used to train the classifier, which was later used to predict the quadrant changes for the complete songs.

Model accuracy was computed as the percentage of correctly classified music clips. Confusion matrices are also provided.

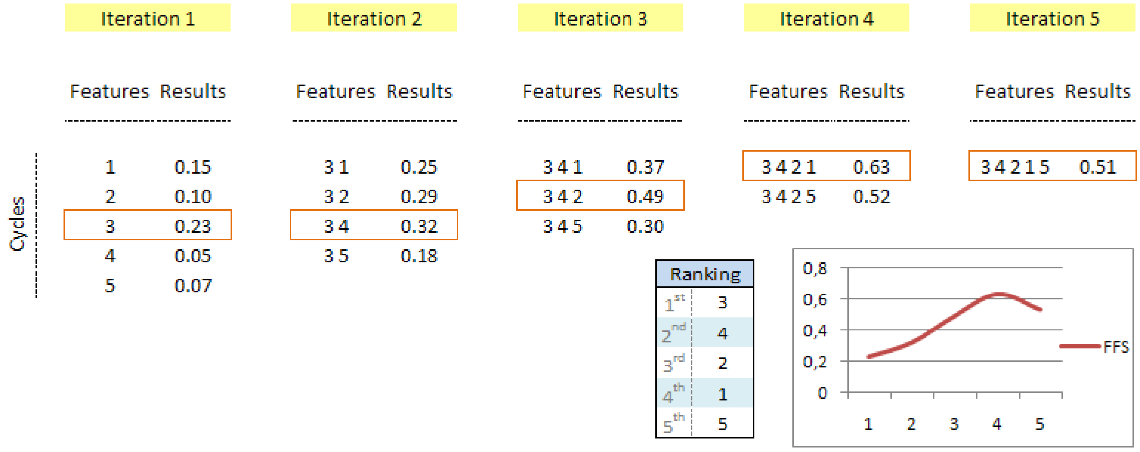
Forward Feature Selection

Feature selection is an important step to improve experimental results. Until know, tests were done with the entire feature set available at the time. However, this introduces the problem of using features that are not interesting to the subject, introducing unneeded data and inducing the classifier in error. There is a need to analyze features relevance and the optimal feature set for our problem.

To achieve this, we used Forward Feature Selection (FFS), a feature selection algorithm that consists in the following steps, starting with an empty "optimal set":

1. Pick one of the remaining features.
2. Train a classifier using the optimal set plus the chosen feature.
3. Test the previously trained classifier and store the results.
4. Repeat for each one of the remaining features.
5. Add the best feature to the end of "optimal set" list.
6. Repeat the entire procedure until no remaining features are left

By using the described algorithm, a ranking of the best features, ordered by relevance, was obtained. The procedure was repeated using different groups of songs for training and testing and the rankings were averaged (Figure 21). The entire FFS algorithm was repeated 6 times and the feature rankings accumulated to form a global ranking.



	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
Cycles	Features	Results	Features	Results	Features	Results	Features	Results	Features	Results
	1	0.15	3 1	0.25	3 4 1	0.37	3 4 2 1	0.63	3 4 2 1 5	0.51
	2	0.10	3 2	0.29	3 4 2	0.49	3 4 2 5	0.52		
	3	0.23	3 4	0.32	3 4 5	0.30				
	4	0.05	3 5	0.18						
	5	0.07								

Ranking	
1st	3
2nd	4
3rd	2
4th	1
5th	5

Figure 21: Forward Feature Selection algorithm

3.1.3. Tracking

Mood tracking implementation was in part similar to the classification process described earlier. The major differences are the usage of a different feature extraction network, as well as the training and classification processes, which were conducted in small song segments (analysis windows) representing intervals of milliseconds, instead of the single vector approach. In addition, the output of each segment is a quadrant instead of an AV pair, as mentioned above.

The size of these small fragments or chunks of data is defined by the number of samples that constitute the analyzed window. Several strategies were approached in order to test and understand the influence of these parameters on the tracking accuracy. For the window sizes, several values, ranging from 512 to 65536 (23 to 2970 milliseconds for songs with a frequency of 22050Hz) were tested to measure the influence of them on the results. Experiments with different memory values for the *Texture Stats*[21] MarSystem were also done. This parameter is set to 40 by default and influences the way features are calculated, serving as a memory, in which the last N feature vectors are summarized.

Other experiment was to compare the influence of window size, e.g., big windows versus averaging the feature vectors of several small windows. For example, calculating the accuracy using the average of 8 1024-sample windows and with an 8192-sample window.

Smoothing

One last step used in our experiments was smoothing or noise reduction. The idea is to ignore mood variations on a tiny time space that may not be significant. This is done by, on a quadrant change, checking its duration, $\boldsymbol{d}$, and the quadrants before, $\boldsymbol{b}$, and after, $\boldsymbol{a}$, the current one. If the quadrants before and after the current one are similar ($\boldsymbol{b} = \boldsymbol{a}$) and the duration, $\boldsymbol{d}$, of the current one is less than the defined threshold, $\boldsymbol{t}$, ($\boldsymbol{d} < t$), then the current quadrant change is ignored.

The results before (Figure 22) and after (Figure 23) applying this process are shown in the pictures below (with $t = 0.5$ seconds).

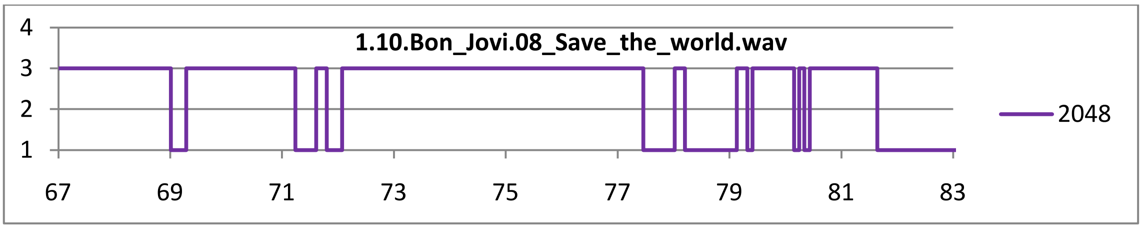


Figure 22: Mood tracking example without smoothing

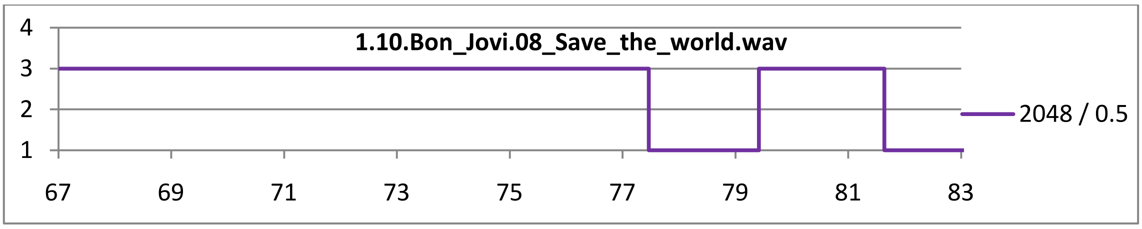


Figure 23: Mood tracking example with smoothing

[21] A texture window analysis MarSystem, used to calculate the mean and standard deviation based on the last *memSize* windows. This composite MarSystem uses the following MarSystems: Series, Memory – a circular buffer that holds the past *memSize* observations, Mean and StandardDeviation.

3.2. Database

An SQL database is used by the server to store all information about the songs and classification. Although managed through the backoffice application, all operations on the database, from querying data to updates and additions of new information is done by the server itself. These operations are conducted using Qt SQL libraries such as *QSqlDatabase* and *QSqlQuery*. Using this simplifies the process of dealing with connectors and provides support for different database management systems. Also, the server will be usable with various engines, from a simple SQLite file to MySQL, PostgreSQL, Oracle, Access DB files or any DB supporting ODBC (Open Data Base Connectivity protocol). On the current prototype only SQLite is supported, with preliminary support for MySQL.

The DB was designed with the future in mind, supporting the current needs but also the ones that will appear in the future, such as various mood models of different types (categorical or dimensional), user accounts, lists of artists, albums, genres, features and classification profiles, saving several classification and tracking values for the same song with different combinations of features and classifiers and others. The current result is represented below with the Entity-Relationship Model, something that might suffer small updates before the application is finished.

Entity-Relationship Model

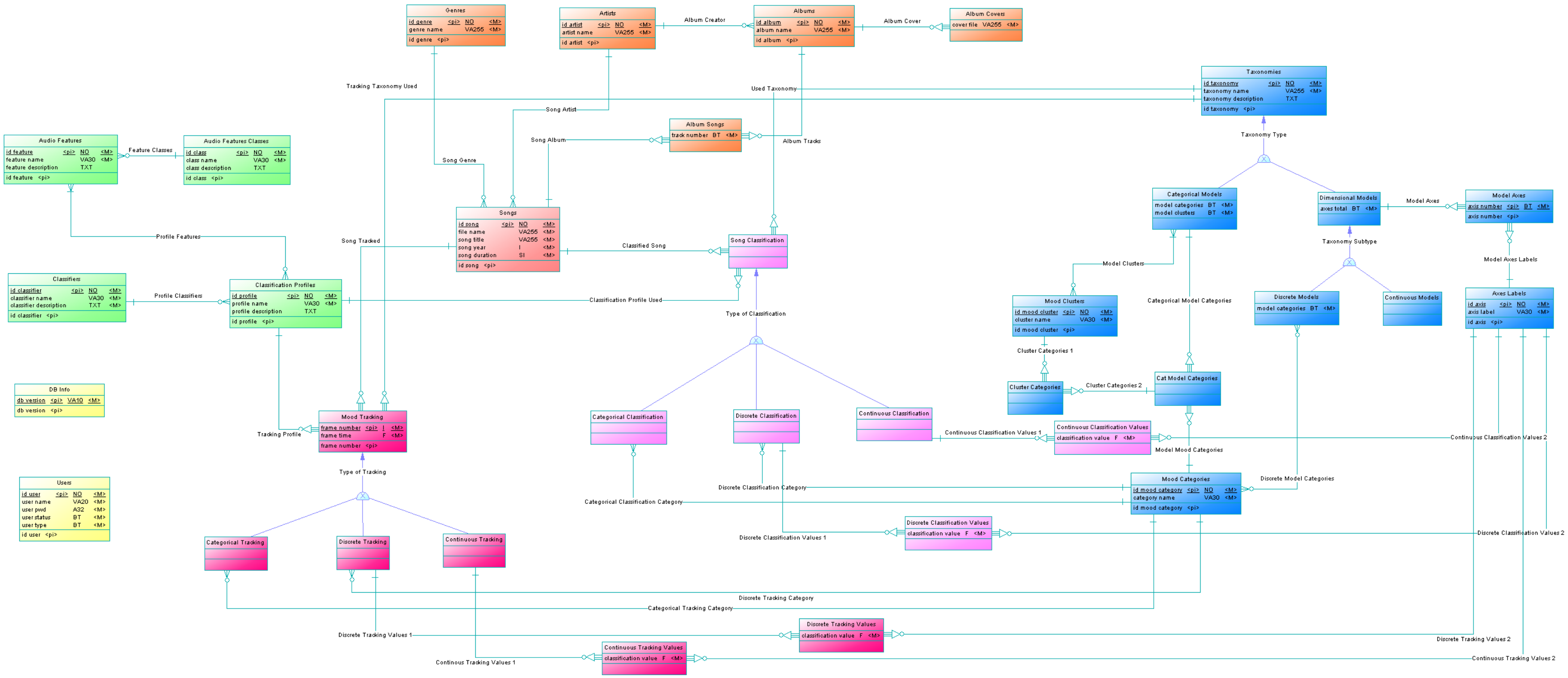


Figure 24: Entity-Relationship Model

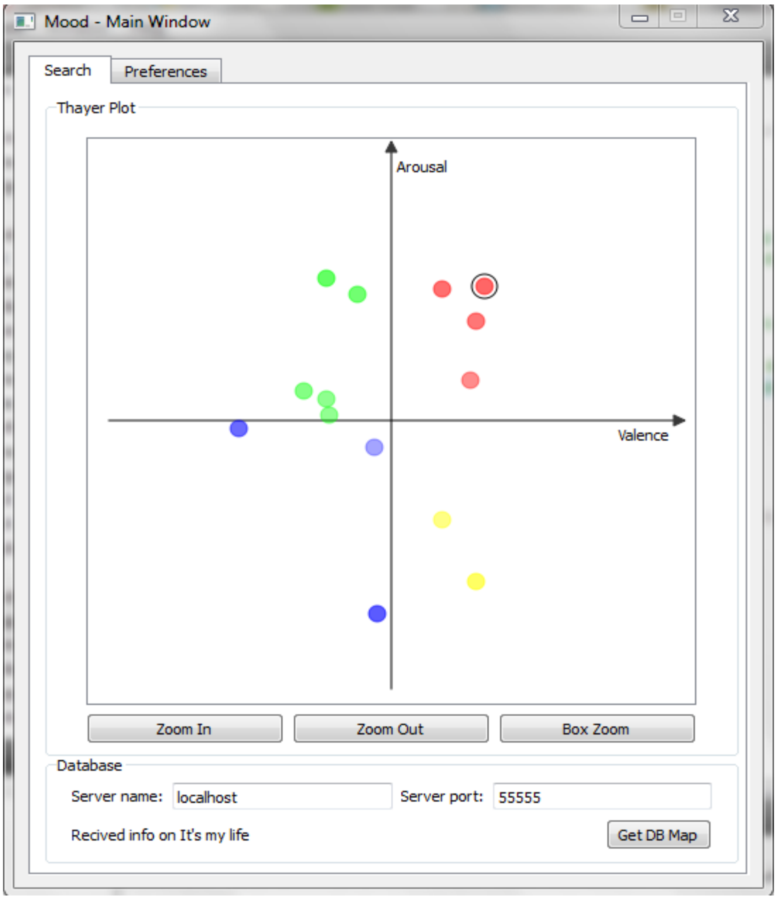
3.3. Client Application

The client application was implemented by another element of the team (Lic. student Luís Cardoso), after the requirement analysis developed by Professor Rui Pedro Paiva, João Fernandes and myself, using the Qt platform. As a result, it is portable and can be used in any of the platforms supported by Qt.

The first version is a prototype, supporting only the features available in the server (the component pertaining to this MSc thesis). Currently it can be used to check the distribution of the songs over Thayer's model, ask for details on each song and also play the selected song, which, at the same time, shows the quadrant changes (highlighted in different colors) in the sound wave graph.

Main Window

Using it is relatively straightforward. After opening the application the user will need to insert the server address and port in order to establish all communications. Once this info is inserted, the DB Map (visual map of all songs in the database) can be requested, as shown in Figure 25. Each point represents a song and the user can use *mouse over* events to check the song name. The user can zoom and interact with the map (draw traces, click songs, etc.). In the future, filter and search options will be also available to help view large databases. The preference settings tab is not available yet, but it will offer the possibility to change settings like requesting the DB Map for different classifiers.



Server name: localhost

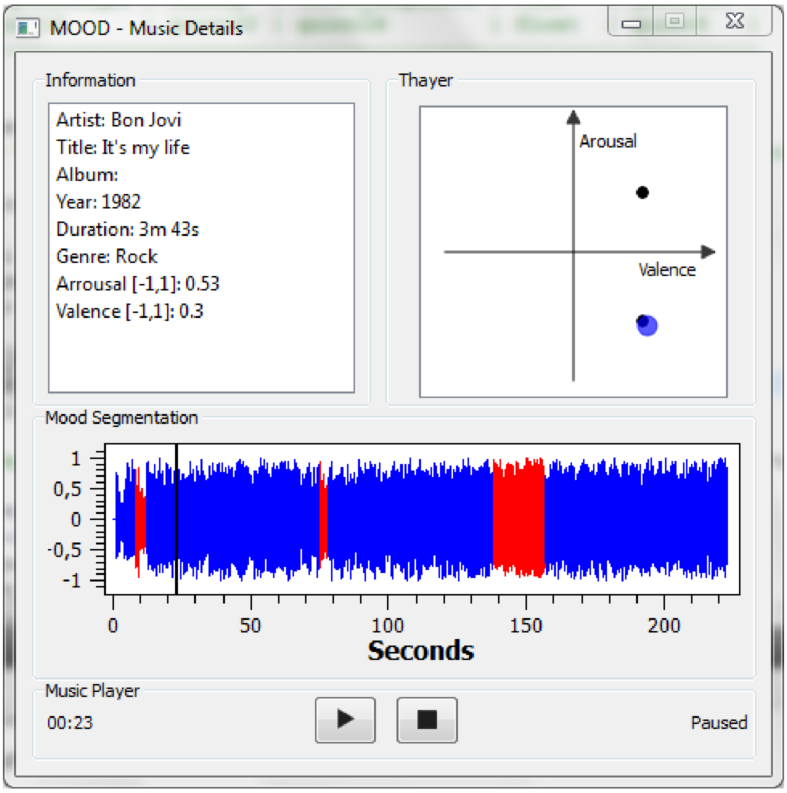
Server port: 55555

Recived info on It's my life

Figure 25: Client Application main window

Song Details

When a song on the DB map is selected a request for the details of that song details is sent. The answer is then parsed and displayed on the *Song Details* form. This form shows all the ID3 tag information stored for that song, as well as the soundwave graph and mood tracking data, if available. The song can then be played and the progress will be marked with a black line on the wave plot, different colors represent different quadrants identified for the current selected profile as demonstrated on Figure 26.



MOOD - Music Details

Information

Artist: Bon Jovi
Title: It's my life
Album:
Year: 1982
Duration: 3m 43s
Genre: Rock
Arrousal [-1,1]: 0.53
Valence [-1,1]: 0.3

Music Player

00:23

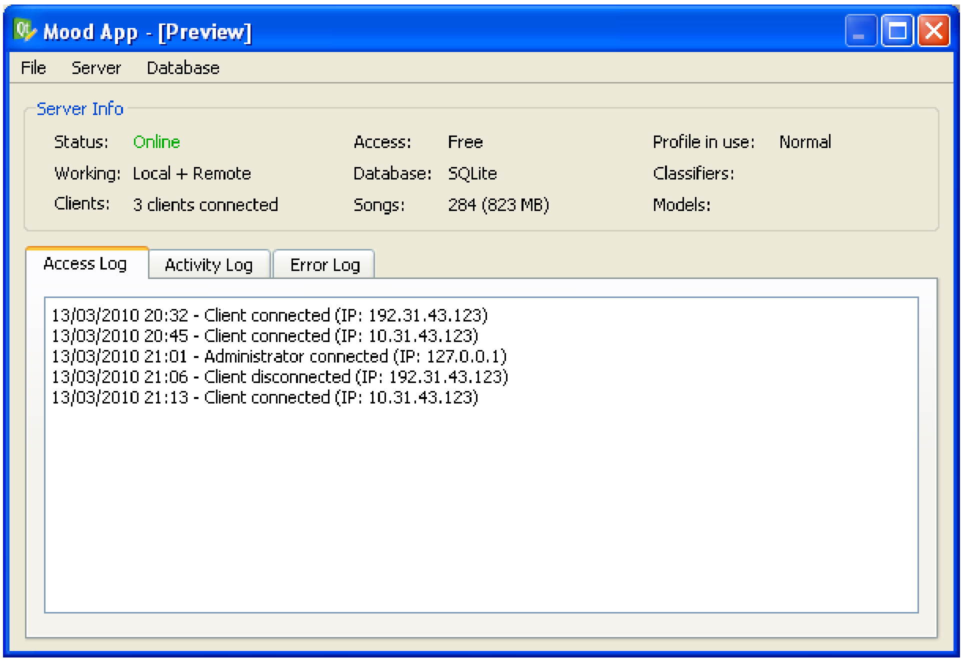
Paused

Figure 26: Client Application Song Details dialog

3.4. Backoffice Application

The backoffice is the application that will serve as a front end to control and administrate the server. It is currently under construction by João, following the requirements developed by the same people as above. Its work mode will be somehow similar to the client application, in that it will work by sending requests and parsing the responses from the server to update its widgets with received data. The implementation of a prototype of this application is not finished, nonetheless a few screen shots of its interface will be listed to give an idea of its work mode and how it will look like.

The first image shows the backoffice main window (Figure 27). After logging in, the user will view this dialog where the main server settings and information is listed. Additionally, access, activity and error logs can easily be accessible in order to guarantee that everything is working as expected.

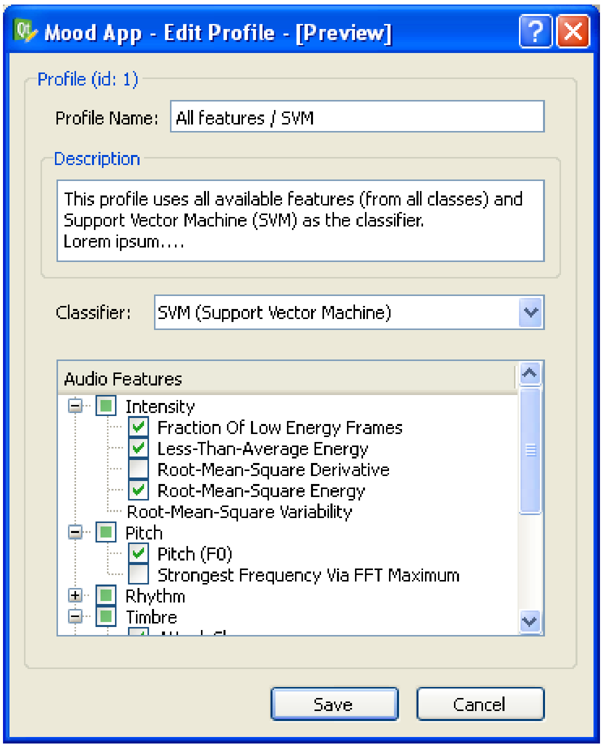


Status: Online
Working: Local + Remote
Clients: 3 clients connected
Access: Free
Database: SQLite
Songs: 284 (823 MB)
Profile in use: Normal
Classifiers:
Models:

```
13/03/2010 20:32 - Client connected (IP: 192.31.43.123)
13/03/2010 20:45 - Client connected (IP: 10.31.43.123)
13/03/2010 21:01 - Administrator connected (IP: 127.0.0.1)
13/03/2010 21:06 - Client disconnected (IP: 192.31.43.123)
13/03/2010 21:13 - Client connected (IP: 10.31.43.123)
```

Figure 27: BackOffice main screen

The following image displays the *Edit Profile* dialog screen (Figure 28). Here the administrator may edit all the parameters that constitute a classification profile, particularly changing features and the classifier. Changing profile settings will make the current classification and tracking records of that profile became invalid. As a consequence all songs need to be processed (or the records dropped).



Profile Name: All features / SVM

This profile uses all available features (from all classes) and
Support Vector Machine (SVM) as the classifier.
Lorem ipsum....

Classifier: SVM (Support Vector Machine)

Audio Features
- ⊟ 🟩 Intensity
  - ☑ Fraction Of Low Energy Frames
  - ☑ Less-Than-Average Energy
  - □ Root-Mean-Square Derivative
  - ☑ Root-Mean-Square Energy
  - Root-Mean-Square Variability
- ⊟ 🟩 Pitch
  - ☑ Pitch (F0)
  - □ Strongest Frequency Via FFT Maximum
- ⊞ 🟩 Rhythm
- ⊟ 🟩 Timbre

Figure 28: BackOffice Edit Profile dialog

3.5. Server Application

The server application answers the queries made by clients, retrieves song information and entire playlists, as well as manages the entire songs and users' database. It can be viewed as various distinct modules: database management, audio processing, client communication and others. In the table below (Table 8) the summary of requirements for the server application is presented (for further details of requirement analysis, see Appendix A):

Requirements	Details
User accounts	- Create account - Remove account - Block / Ban account - Edit profile - Encrypt sensible user information (password at least)
Client communication	- User authentication - Process user queries - Return query results (m3u, song info, …) - Stream/send songs - Receive songs from users/administrators - Remove song from DB - Change server settings (administrators)
Database management	- Create database - Drop database - Delete database - Insert new songs information - Update / edit existent information - Select / browse songs information
Audio processing	- Down sample songs - Extract audio features - Apply classifiers

Table 8: Server application requirements

We have implemented a prototype version of the server application. Currently, this version serves as a concept of how the entire system will work in the future (recall that playlist generation is part of another MSc thesis, which is still under work), showing part of the desired functionalities.

The main functions of the server are to communicate with the client and backoffice applications, as well to process audio files and interact with the database. Although it does not have a GUI, the *MOOD Server* is built using the Qt framework. The main reason for this is the support from the framework for non-GUI features like SQL database access, XML parsing, thread management, network support, and a unified cross-platform API for file handling, great advantages when building what should be a cross platform application, as well as the integration with the other Qt-based applications (Client and backoffice).

Communications

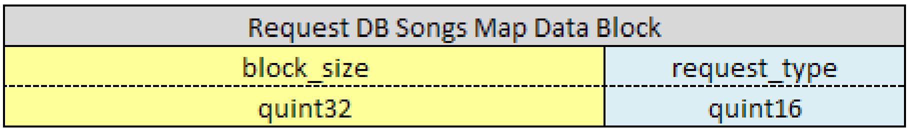
To accept connections from both types of users, the server has two independent sockets running in different ports, one for the client application and the second for the backoffice, by implementing a subclass of the *QTcpServer* class. Both use threads to handle requests from different clients simultaneously, creating a new one to respond to each client connection (and request).

Messages are exchanged through a *QTcpSocket*, where clients are expected to send requests immediately after establishing a connection. A custom block-oriented protocol is used to transfer data as binary data blocks through a *QDataStream* instead of a *QTextStream*. Each block consists of a size field followed by that amount of data. All the data blocks must start with a parameter identifying their type of request or response in order to be correctly processed. The remaining data depends on this parameter. Both client and server need to know and use all the blocks being sent and received in order to correctly parse the data that comes in the form of an array of bytes (*QByteArray*). Some of the used data blocks are described below.

Data Blocks

All used blocks start with the size of the block (quint32) in order to know how much data we should receive before parsing the message, followed by a number (quint16) that identifies the message type.

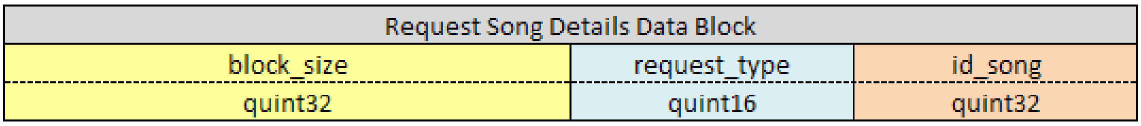
A data block similar to the image above (Figure 29) is received by the server when a client asks for the database map, where the field *request_type* will have the value 1.



Request DB Songs Map Data Block	
block_size	request_type
quint32	quint16

Figure 29: Request DB Map data block

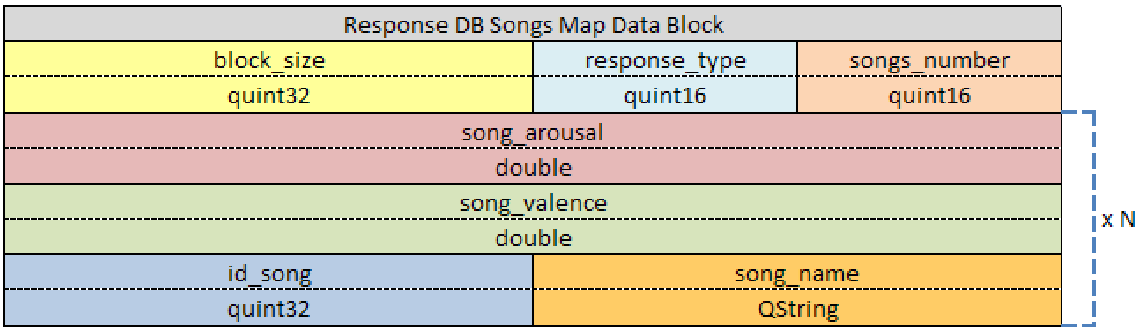
A request for song details is represented by the block below (Figure 30). Here, the *request_type* will have the value 2. The *id_song* parameter, as the name indicates, is the id of the song to query.



Request Song Details Data Block		
block_size	request_type	id_song
quint32	quint16	quint32

Figure 30: Request Song Details data block

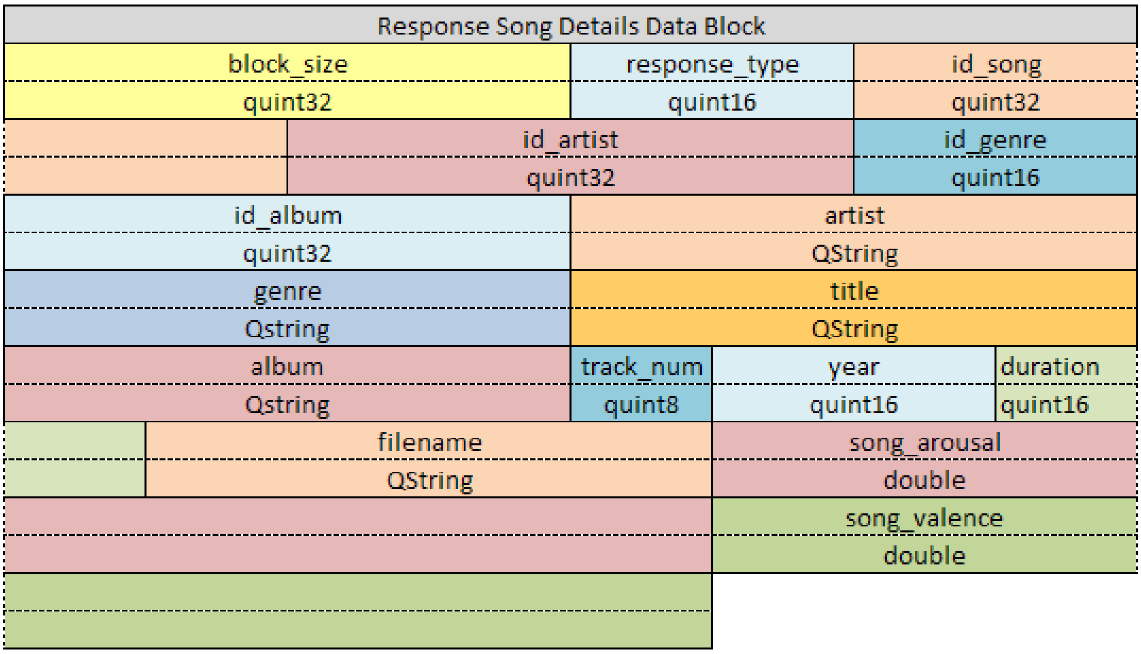
After receiving a request for the DB map, the server will query the database and return the following data block (Figure 31). In this block, *response_type* will be 1. The *songs_number* parameter represents the total number of songs that are described. The rest of the block has song data (id, name, arousal and valence) repeated a total of *songs_number* times.



Response DB Songs Map Data Block			
block_size quint32	response_type quint16	songs_number quint16	
song_arousal double			x N
song_valence double			
id_song quint32	song_name QString		

Figure 31: Response DB Songs Map data block

The block to send details of a given song is presented in Figure 32. All the field names are clear about what they represent. The *response_type* field will be 2.



Response Song Details Data Block			
block_size quint32	response_type quint16	id_song quint32	
	id_artist quint32	id_genre quint16	
id_album quint32	artist QString		
genre Qstring	title QString		
album Qstring	track_num quint8	year quint16	duration quint16
	filename QString	song_arousal double	
	song_valence double		

Figure 32: Response Song Details data block

Settings

All the configuration parameters of the server are stored on an INI file (settings.ini) in a human readable format so they can also be easily edited. The settings stored go from the ports and network interfaces used (which define if the server works only at *localhost*, LAN or Internet), all database settings, path to the songs directory and so on.

The file is stored using *QSettings* class, which provides a persistent platform-independent way of storing application settings. This information is often stored in the system registry on Windows, and in XML preferences files on Mac OS X. On Unix systems, in the absence of a standard, many applications (including the KDE applications) use INI text files. To avoid writing to the windows registry, we chose to use an INI file for all systems.

Finally, an extra detail is the usage of Signals and Slots to communicate between objects, something that is usually done in other frameworks by using callbacks. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks. A signal is emitted when a particular event occurs and a slot is a function that is called in response to a particular signal. One example of this in our project is the *incomingConnection()* slot, that is activated by *QTcpSocket*, which emits a signal every time a new connection is established.

4. Experimental Results

After having implemented the audio analysis and processing logic described in the previous chapter, various groups of tests were conducted, assessing many different parameters for tracking and classification in order to obtain results that would allow us to evaluate the system accuracy and under which parameters and circumstances it performs best.

4.1. Annotations

Annotations are one of the bases of this work. They have crucial importance on the results, as they are used in the classifier training process and to measure the results. In this work we used arousal and valence annotations kindly provided by Yang [Yang et al., 2008] for training and classification testing. To test and evaluate the mood tracking, a different set of quadrant annotations, made by two volunteers, was used.

4.1.1. Yang Annotations

These arousal and valence annotations were made from 25 seconds clips that better expressed the emotion present on each song, for a total of 194 songs. The songs were selected by Yang, trying to have a balanced number of songs for each of the four existent quadrants (48 to 49). However, there are some problems with them that may have a negative influence on the results.

Proximity to the origin of the graph

One of the main drawbacks with Yang annotations is their distribution over the Thayer's plan. According to the same model, emotions are always placed near the outside of the graph, where the reference values are relevant, with a high positive or negative valence and arousal.

Nevertheless, placing these annotations on the Thayer's model of mood reveals that a high number of songs are close to the graph origin as exemplified in Figure 33. Here, we can see a great accumulation of annotation points near the center, with 47 within a distance of 0.25 (inside the red circumference) and 140 within 0.50 (orange circumference) as listed in Table 9.

Norm	Songs	Sum
[0 , 0.25]	47	47
]0.25 , 0.5]	93	140
]0.5 , 0.75]	47	187
]0.75 , 2]	7	194

Table 9: Yang Annotations – distance to the origin

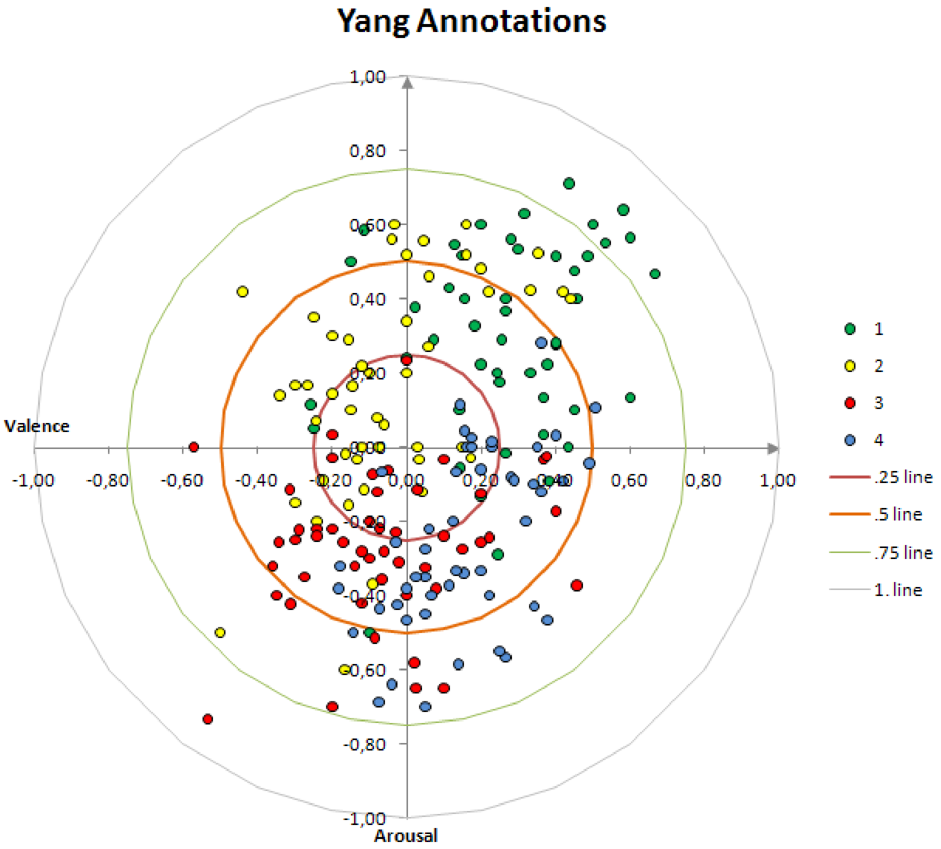


Figure 33: Yang Annotations placed on the Thayer's model

Unbalanced Song Distribution

One of the concerns taken into account by Yang when compiling his test collection was to have an equal number of songs for each quadrant. Therefore, he picked 48 to 49 songs for each of the four quadrants based on his opinion and feelings. However, the annotations demonstrated that in many cases the songs do not belong to the initial quadrants in subjects' opinion, thus making the distribution unbalanced. This problem is demonstrated in the picture (Figure 33). The four different colors used represent the four different quadrants. It would be expected, according to Yang division, to see points with the same color grouped in the same quadrant. Instead all colors are scattered between all quadrants, with the second having less songs than any other as detailed in Table 7, repeated here for simplicity (Table 10). In the refereed table, “Other” represents songs that are exactly placed between two quadrants, by having a value of zero for arousal or valence. “Match” represents the number of songs placed by Yang that were indeed correct.

Quadrants	Yang	Annotation	Annot (%)	Match	Match (%)
1	48	54	27.84%	36	75.00%
2	48	22	11.34%	17	35.42%
3	49	51	26.29%	16	32.65%
4	49	49	25.26%	9	18.37%
Other	0	18	9.28%	-	-
Total	194	194	100.00%	78	40.21%

Table 10: Songs per quadrant: Yang vs. real annotations

4.1.2. Tracking Annotations

To evaluate mood tracking results, the existing annotations were not enough, since we needed details about mood changes during the entire songs. To solve this problem, two volunteers listened to 57 full songs of the 194 (oriental songs were excluded) and registered changes between quadrants for the entire songs duration. The new annotations were analyzed and compared in order to measure the matching ratio between volunteers. To execute our experiments only 29 songs, where volunteers agreement was higher than 80%, were used.

The tracking annotations collected by us also have some flaws that were exposed during tests.

Number of Subjects

The annotations are not really significant of a population since they were created by a really low number of persons – two. With so few annotations there is no way to properly distinguish when there are clear mood changes, where most of the subjects would agree and few disagree, or when some songs don’t generate consensus at all and all subjects disagree. The number of subjects is insufficient to have a clear majority even when a few disagree.

Unbalanced Song Distribution

Similar to Yang’s situation, the selected songs were mostly from the first quadrant according to Yang´s annotation, as described in Table 11. The optimal scenario would be to test the tracking algorithms with a group of songs equally distributed to verify the accuracy for each one of the four existent quadrants.

Quadrants	All	Match +80%
1	23	11
2	13	6
3	10	6
4	5	2
Other	6	4
Total	57	29

Table 11: Mood tracking annotations distribution according to Yang

Differences between tracking and Yang annotations

One last problem present with the mood tracking annotations is that, in some cases, there seems to be an inconsistency when compared to Yang's annotations, which cannot be further investigated due to the small population (and annotations) available.

As explained, Yang's annotations were done by at least 10 subjects for each song, evaluating 25 seconds clips that better represented each one. Given that, it would be expected that each mood tracking annotation had segments that matched Yang's annotation. However, this does not always verify, with several tracking annotations alternating between two different quadrants but never coinciding with Yang's values, as exemplified in Figure 34. In this example, the blue line represents Yang's annotation. It indicates that, generally, the 10 subjects used by Yang placed the clip in the second quadrant (e.g. angry), with negative valence (although very close to zero). However, the subjects responsible for the mood tracking annotations only detected segments of quadrants one (e.g. happy) and four (e.g. peaceful), never finding a negative valence in the entire song. Although this is possible, due to the subjectivity of the topic or cultural reasons, it still highlights one weakness of our process, already related above – the low number of subjects used.

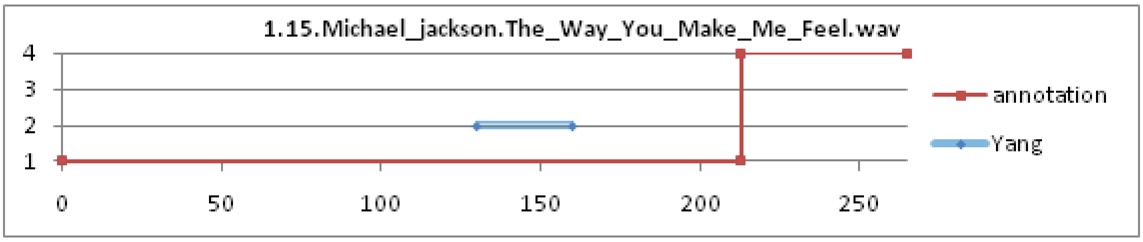


Figure 34: Annotation inconsistencies: mood tracking vs. Yang

4.2. Global Classification

For global classification 20 repetitions of 10-fold cross validation tests were run. This ensures that all songs are used in different groups for training and testing. The final results for AV (SVR) are the average of all repetitions. Regarding quadrants classification (SVC), the results represent the quadrant predicted the most for each song. In the feature extraction process, 512 samples windows were used with a memory value of 40.

4.2.1. AV (SVR)

To analyze the classification results, R2 statistics and RMSE were the choice (see 3.1.2. – Classification).

Using all available features

Tests using the entire feature set confirmed that arousal is in general easier to predict than valence, a fact confirmed by the *R2* statistics. The *R2* values reached 57.9% for arousal and 3.24% for valence as shown in Table 12. Arousal results are similar to ones the observed in [Yang et al., 2009] but the valence value for *R2* is way below the result of 28.1% from the same paper. The observed discrepancy for valence is a result of the feature set available in each experiment and the group of selected features to be used. Yang worked with several feature extraction frameworks, obtaining a wide range of features that are not available in Marsyas at this time. This affects specially valence since from the four most important features mentioned in [Yang et al., 2008], Marsyas lacks three of them (Spectral Dissonance, Tonality and Chord). We conducted a pilot study with the MIR Toolbox, which led to an R2 value of 25% for valence, thus confirming the lack of meaningful features in Marsyas. The second cause is related with the use of the entire feature set. Using all features may be decreasing the results especially for valence and reducing the accuracy of the model due to features that do not apply to the context of mood analysis.

arousal				valence			
SSE	SST	RMSE	R2	SSE	SST	RMSE	R2
0.966816	2.34311	0.220108	0.57985	1.15312	1.1943	0.241297	0.0324472

Table 12: Global classification results (all features)

Looking at the placement of the predictions on Thayer's model shows that all songs are gathered within the 0.50 circumference. It is also clear that the values predicted for valence have a very small variation (Figure 35).

Suggestions to improve these results are:

- Add meaningful features, namely tonality, multiplicity, spectral dissonance and chord mentioned in [Yang et al., 2008].
- Use a feature selection algorithm (FFS), selecting the optimal feature set for both arousal and valence.
- Experiment with different classification parameters and settings and check the importance of parameter selection for *libSVM* is described in [Hsu et al., 2010].
- Better annotations, balanced and far from the origin.

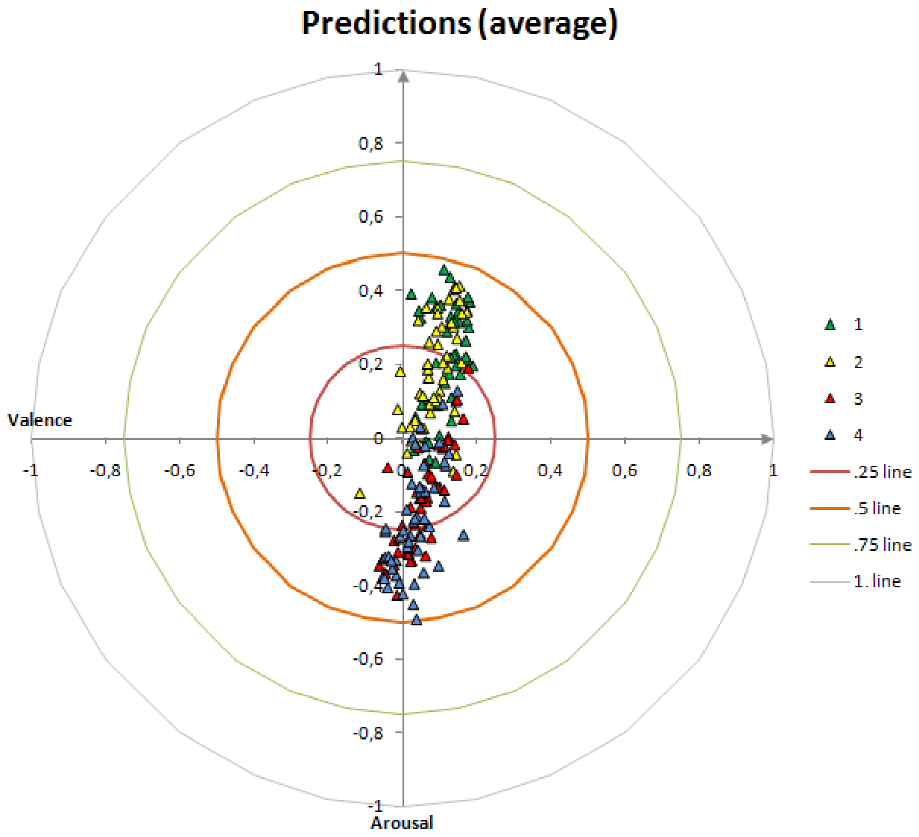


Figure 35: Global predictions in Thayer's model (all features)

4.2.2. Quadrants (SVC)

The accuracy of these tests was measured with the number and rate of correct predictions. Since no regression was used there are no metrics to measure wrong predictions, only if the predicted classes (in this case, quadrant) were correct.

All Features

The results obtained in this experiment were generally weak, considering the low granularity in use (only 4 classes). An accuracy of 47.42% was obtained, with 92 correct predictions in a total of 194 songs. No songs were classified as from quadrant 2 and 4. A fact that may be attributed in part to the unbalanced data set, especially for quadrant 2. The predictions distribution is showed with more detail on the confusion matrix (Table 13).

One reason for this result was probably the usage of all features, where many may not be useful for the subject lowering the accuracy of the classifier.

Quadrants		Annotations				
		1	2	3	4	
Predicted	1	54	21	16	17	108
	2	0	0	0	0	0
	3	12	1	38	35	86
	4	0	0	0	0	0
		66	22	54	52	

Table 13: Confusion matrix (SVC, all features)

Forward Feature Selection

After testing with the entire feature set, the FFS algorithm described in chapter 3.1.2 was applied to calculate de features ranking, using 10% of the collection for testing. This process was repeated 6 times and the final ranking resulted in a list of features ordered by relevance.

Following, the list was used to find the optimal number of features – the list which provided the best results. This was done by training and testing a classifier (using cross fold validation), starting with only the top feature and continuously adding the next feature, registering the results for each iteration, until all features were tested.

The results indicated that the best value for accuracy, 48.95% or 95 correct predictions of a total of 194, occurred when using the top 383 features (Figure 36). This final list was then used to predict the quadrant for each song, using a 10-fold cross validation system and the most predicted quadrant for each song was saved.

Comparing the confusion matrix from FFS (Table 14) with the previous one, referent to all features, we noted that few things changed, with only two incorrect predictions of the third quadrant songs being predicted correctly this time.

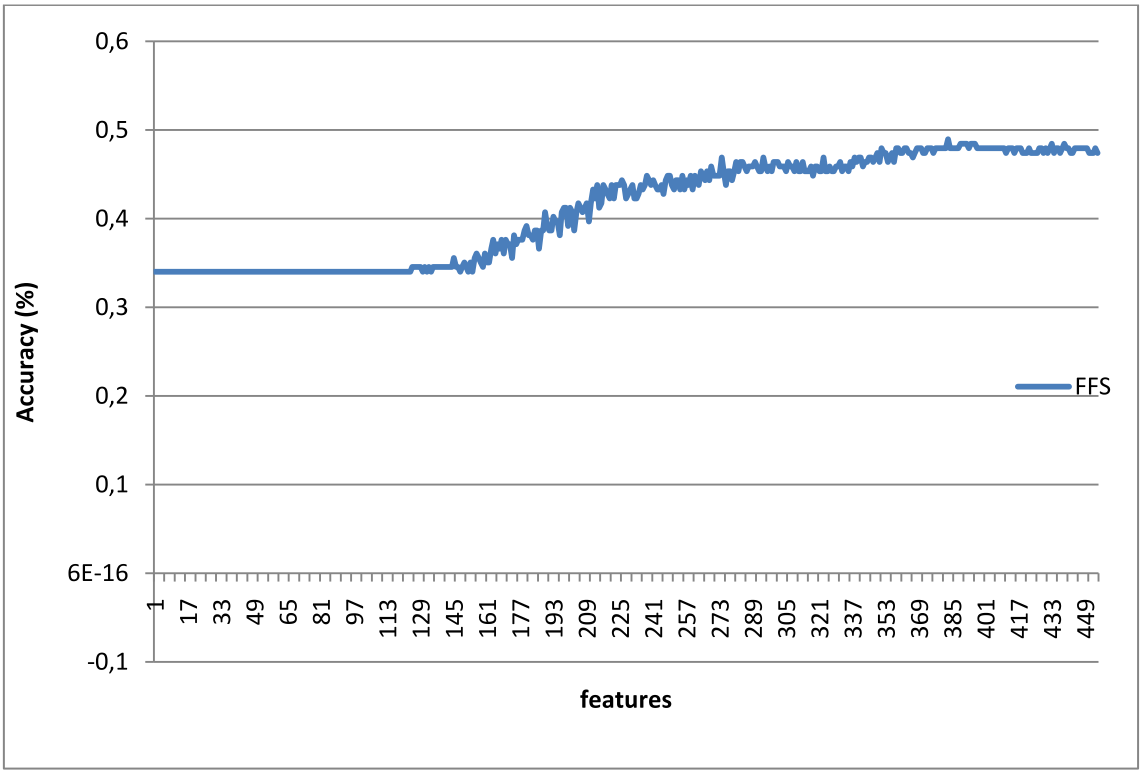


Figure 36: Features accuracy based on FFS feature ranking

Quadrants		Annotations				
		1	2	3	4	
Predicted	1	54	21	14	17	108
	2	0	0	0	0	0
	3	12	1	40	35	88
	4	0	0	0	0	0
		66	22	54	52	

Table 14: Confusion matrix (SVC, top 383 features)

By analyzing the results obtained, we can speculate some probable causes for of the low accuracy and especially the weak FFS results. First of all, using the FFS with classification instead of regression means that the results will have less variation. For a small test set (composed of 20 songs), the results obtained with two features, even if one is superior, may not be enough to increase the accuracy by predicting correctly one more song, in contrast with using R2 values. These cases and also features with little relevance will result in many similar matching percentages for many features (e.g. 10/20), something that was verified in our test causing the first feature of this similar results list to be picked, while it may not the best of the set. Apart from the previous problem, the most obvious reason for the verified outcome is the unbalanced nature of our annotations. The set is clearly unbalanced in favor of the first quadrant, with 66 songs, 12 more than the second with most songs. This fact will create an unbalanced SVC model and any feature, even if with little or no relevance, will have a naturally higher probability of being predicted with the quadrant 1 over any other. The described result is clearly confirmed by the line on the above graph, marking a result of 0.34 for the first 100 features. It was verified that, at the beginning of FFS, the unbalanced model predicts the first quadrant to all songs with most if not all of the features, guaranteeing a result of 66/194. This continues and only after more than 100 features were added the results start to change, with predictions of third quadrant songs.

With the optimal feature set of this experiment (383), it was also verified that all the predictions are between quadrants 1 and 3, with quadrant 4 being predicted few times during the cross validation and no predictions for quadrant 2, the one with the least songs.

To improve the experiment, a bigger, balanced data set must be used, with an equal number of songs for each quadrant. Nevertheless, the experiment would probably be more relevant using AV values and R2 as measure.

4.2.3. Tracking

For mood tracking the results are based on the matching percentage between the songs predictions and the respective annotation.

All Features

The results for mood tracking when using all features were varied. For mood tracking several settings were tested as described previously in the “Implementation” chapter. Of these settings, window size in samples showed to be the most relevant one, followed by smoothing values.

The best result obtained was 44.09%, using windows of 2048 samples. In the same way to what happened previously, we verified excellent results with songs when the tracking annotation was mostly in quadrant 1. On the other hand, the accuracy dropped to really low values every time the annotations marked quadrant four but especially quadrant two, since it was never predicted.

Similarly to what was suggested for global classification, there are several points where results can be improved:

- A balanced data set that will allow a correct SVC training and also a well calculated feature ranking.
- Improved tracking annotations, done with more subjects and over a higher data set.
- Extra audio features, that showed to be good in mood classification and tracking problems, according to the literature such as [Yang et al., 2008].

Matching results are available in Table 15. There, for different values of windows sizes in samples, the percentage of matching for each of the 29 tested songs is indicated. The last row indicates the average of all songs.

With these results we can conclude that smaller windows sizes tend to achieve a better matching percentage, with 2048 samples demonstrating to be the best value, as stated earlier. Both the discrepancy between obtained results and the average matching values could be higher with a better trained classifier and by using only the best features.

	Mood Track Matching (winSize) mem = 40						
Song	1024	2048	4096	8192	16384	32768	65536
1	91.41%	90.98%	91.35%	92.09%	85.06%	91.64%	91.64%
2	90.47%	90.47%	90.47%	90.47%	90.32%	90.02%	89.43%
3	52.69%	53.63%	53.09%	50.79%	51.31%	50.30%	55.28%
4	89.57%	88.63%	89.39%	85.09%	82.22%	89.33%	89.33%
5	80.09%	80.07%	80.07%	80.07%	80.07%	80.07%	79.63%
6	76.84%	76.32%	74.85%	72.04%	81.54%	78.51%	78.51%
7	79.90%	79.87%	80.49%	79.87%	79.87%	79.27%	52.64%
8	95.40%	97.15%	97.15%	97.15%	94.44%	93.87%	86.04%
9	85.07%	85.07%	85.07%	85.07%	85.09%	85.09%	85.09%
10	91.85%	92.18%	92.18%	92.18%	92.18%	92.18%	91.92%
11	7.11%	5.58%	7.17%	7.75%	7.75%	4.30%	0.00%
12	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
13	5.89%	4.50%	2.42%	2.37%	1.28%	0.00%	0.00%
14	0.75%	0.88%	1.13%	1.13%	0.00%	0.00%	0.00%
15	45.70%	41.03%	40.86%	54.08%	38.75%	44.56%	53.89%
16	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
17	45.75%	39.94%	27.39%	48.62%	53.82%	70.73%	79.27%
18	0.24%	0.24%	1.95%	4.60%	9.67%	0.00%	14.15%
19	27.08%	23.52%	25.45%	38.70%	26.02%	33.45%	19.00%
20	3.08%	1.52%	3.40%	0.00%	0.00%	0.00%	0.00%
21	37.75%	51.12%	57.06%	44.78%	14.91%	0.00%	0.00%
22	56.45%	58.44%	41.21%	53.73%	53.09%	54.49%	94.38%
23	40.21%	52.42%	74.17%	26.41%	0.00%	0.00%	0.00%
24	3.57%	3.06%	2.50%	0.00%	0.00%	0.00%	0.00%
25	55.67%	60.58%	58.22%	45.82%	38.66%	49.47%	26.14%
26	6.90%	5.29%	2.70%	2.54%	0.00%	0.00%	0.00%
27	19.97%	19.29%	18.55%	17.61%	13.31%	1.99%	0.00%
28	30.56%	30.61%	26.15%	12.00%	8.00%	0.53%	1.06%
29	45.30%	46.35%	48.23%	39.52%	37.05%	19.13%	13.39%
Average	43.63%	44.09%	43.89%	42.22%	38.77%	38.24%	37.96%

Table 15: Mood tracking results for different windows sizes

5. Conclusions

At the end of the year we believe that most of the objectives for this project were achieved. Important research on MIR, MER and mood tracking was made and, although there is still work to be done, a functional prototype version of our mood application was produced.

One of the most important accomplishments of the project was the knowledge acquired from the research, which happened during the entire year, but also from developing the application and experimenting with the different frameworks. A gradual learning process, as new ideas required a deeper knowledge about the frameworks, especially Marsyas.

The main characteristic of this project was its diversity, covering vast areas of research but also interesting fields of software engineering as described in this thesis. The result is this work. The knowledge gathered, as well as the problems and solutions found during the year, are described here and serve as a valuable resource for future projects on the same field.

Some ideas did not evolve as initially intended or did not attain the initial goals. Things such as a fully usable mood application and the classification and tracking system were not completed but the topics were analyzed in the dissertation, and the path is open to properly achieve these goals in the near future.

Concluding, the project as a whole was a worthy experience and the results will be valuable to the evolution of the mood application and, therefore, to enrich the Department's legacy.

5.1. Future Work

As described along the entire dissertation, the future work should be focused on improving the results of the classification and tracking system as well as concluding the mood server, client and backoffice applications.

To improve the experimental results there are several points that should be taken into consideration:

- A bigger, balanced data set for classification with AV values, with these values being away from the origin.
- Experiment and tuning the various settings on the classifiers.
- Correctly usage of FFS, resulting in a lower number of high relevance features for each subject (quadrants, arousal, and valence).
- Mood tracking annotations for a higher number of songs and, especially, done by more than two volunteers in order to be relevant.

- Addition of extra audio features, by implementing new ones on Marsyas or by adding support for different frameworks.

It is our belief that by accomplishing some of the points listed here, a rise in the results will almost surely be seen.

6. Bibliography

Bray, S., & Tzanetakis, G. (2005). Implicit Patching For Dataflow-Based Audio Analysis And Synthesis.

Carvalho, V., & Chao, C. (2005). *Sentiment Retrieval in Popular Music based on Sequential Learning.*

Cheveigné, A. d., & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am. , 111*.

Feng, Y., Zhuang, Y., & Pan, Y. (2003). *Music Retrieval by Detecting Mood via Computational Media Aesthetics.*

Hevner, K. (1936). Experimental studies of the elements of expression in music. *American Journal of Psychology , 48*, 246-268.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010, April 15). A Practical Guide to Support Vector Classification.

Kleinginna, P. R., & Kleinginna, A. M. (1981). A categorized list of emotion definitions, with a suggestion for a consensual definition. Motivation and Emotion.

Korhonen, M., Clausi, D., & Jernigan, M. (2006). Modeling Emotional Content of Music Using System Identification. *IEEE Transactions on Systems, Man and Cybernetics , 36*, 588- 599.

Laar, B. v. (2006). *Emotion detection in music, a survey.*

Lartillot, O. (2009). *MIRtoolbox 1.2 User's Manual.*

Lee, K., & Slaney, M. (2007). A Unified System For Chord Transcription And Key Extraction Using Hidden Markov Models. *Austrian Computer Society* .

Li, T., & Ogihara, M. (2004). *Content-based Music Similarity Search and Emotion Detection.*

Li, T., & Ogihara, M. (2003). *Detecting Emotion in Music.*

Liu, D., Lu, L., & Zang, H.-J. (2003). *Automated Mood Detection from Acoustic Music Data.*

Lu, L., Liu, D., & Zhang, H.-J. (2006, January). Automatic Mood Detection and Tracking of Music Audio Signals. *IEEE Transactions on Audio, Speech and Language Processing , 14*.

McEnnis, D., McKay, C., Fujinaga, I., & Depalle, P. (2005). jAudio: A Feature Extraction Library.

McKay, C. (2005). jAudio: Towards a standardized extensible audio music feature extraction system.

Meyers, O. C. (2007). *A Mood-Based Music Classification and Exploration System.*

Paiva, R. P. (2006). *Melody Detection in Polyphonic Audio.*

Percival, G., & Tzanetakis, G. (2009). *Marsyas User Manual.*

Tellegen, A., Watson, D., & Clark, L. (1999). On the dimensional and hierarchical structure of affect. *Psychological Science* .

Thayer, R. E. (1989). The biopsychology of mood and arousal. *Oxford University Press* .

Tzanetakis, G. (2002). *Manipulation, Analysis and Retrieval Systems For Audio Signals.*

Yang, D., & Lee, W. (2004). *Disambiguating Music Emotion Using Software Agents.*

Yang, Y.-H., Lin, Y.-C., Su, Y.-F., & Chen, H. H. (2008, February). A Regression Approach to Music Emotion Recognition. *IEEE Transactions on Audio, Speech, and Language Processing , 16*, pp. 448-457.

Yang, Y.-H., Liu, C.-C., & Chen, H. H. (2006). Music emotion classification: A fuzzy approach. *Proc. ACM Multimedia*, (pp. 81-84).

APPENDIX

Appendix A – Requirements Analysis

In any software project, identifying requirements is a natural and crucial stage to the development process. When planning an application, it is important to understand its purpose, the target audience and the needed features, as well as the unique ones that will distinguish the application from the others.

Brainstorming, imagining hypothetical scenarios and making requirements analysis based on them is a good starting point to understand application needs and uses. This analysis is valuable indicating the features needed. Obviously it is not required to have a full detailed description of each feature and action but it is vital to identify the key features that will compose the application.

Application Overview

Briefly, the aim of the project is to develop a system able to correctly identify mood in musical pieces. The application will have a database, storing information about songs mood for the entire piece and how it changes over time. This information can be managed and new songs can be analyzed and added by administrators. Users will use the application to obtain mood information from the database or from their own songs. Among other things, it will be possible to observe mood tracks in real time (in Thayer's plan) and visualize the song sound waveform, highlighting each quadrant in different colors. Other functionalities such as creating playlists based on a song or point on the plan will also be available and are part of João's work.

After the initial brainstorming, a few concepts started to take shape and some requirements and ideas about system architecture were defined. Given the existing needs, it became clear that the system must be composed by three different applications, working in a client / server architecture. The server will be responsible for all the audio processing, feature extraction and database management logic, extracting and analyzing the audio files and to create and manage the songs and users database with the returned information. The server will also listen for communications from clients, in order to receive administration instructions to manage the database or just to return and browse existing information.

The clients' role is to communicate with the server, serving as a frontend to it. The client application will be the interface for users to browse information and to access related functionalities. The administration application serves as a frontend to the administration and maintenance of the server. It can be used locally or remotely to control all operations related with database management, music processing and uploading of new files. Even if performed in the server, these functionalities are all commanded through an administration client, requiring correct credentials in order to protect data.

Next, the requirements analysis for the previously presented architecture will be detailed. These lists are important to identify the main sections of the application. It is also possible to have some requirements that will not be implemented, but they should be kept in the analysis for future reference.

This brief application description is summarized in Figure 39 (Chapter 4).

Server Application

As discussed above, the server application will answer to the queries made by clients, retrieve songs information and entire playlists as well as manage the entire songs and users' database. It can be viewed as various distinct modules: database management, audio processing, client communication and others. In the table below (Table 16) the requirements analysis for the server application is presented:

Requirements	Details
User accounts	- Create account - Remove account - Block / Ban account - Edit profile - Encrypt sensible user information (password at least)
Client communication	- User authentication - Process user queries - Return query results (m3u, song info, …) - Stream/send songs - Receive songs from users/administrators - Remove song from DB - Change server settings (administrators)
Database management	- Create database - Drop database - Delete database - Insert new songs information - Update / edit existent information - Select / browse songs information
Audio processing	- Down sample songs - Extract audio features - Apply classifiers

Table 16: Server application requirements

Client Application

The client application is used by normal users to communicate with the server. Through it, users can create an account, browse DB information by searching songs with specific mood, receiving song details, mood tracking information, recommended playlists and even the songs themselves.

A detailed list of requirements for this application is presented below in Table 17

Requirements	Details
User credentials	- Register user - Login - Logout - Edit profile / change password
Server communication	- Connect to server (local / remote) - Transmit queries - Receive/process results - Receive / download audio file or stream -Send song to be processed
Browse information	- View all DB songs in Thayer's model (DB map) - Create playlists by: - selecting/uploading a song file - a point in the AV 2D graph - playlist trajectory - selected area in the AV 2D graph - Apply zoom to the DB map - Filter map view by: - musical genre - artist - album - year (or year interval) - other relevant musical attributes - a selected group of audio features - View mood tracking information (graph) for a database song - View mood tracking in real time (graph tracing) - View mood tracking information in wave form graph - Other methods of viewing results - Export playlist (m3u) - Exclude songs from suggested playlist - Download songs

Table 17: Client application requirements (Client Application)

Administration Application

The administration application serves as a frontend to manage servers and their settings. Among some functionalities, administrators are able to add and remove songs, update existent songs and users' information, changing server settings and managing users related with audio processing or the server operation itself.

A detailed list of administration module requirements is presented below in Table 18.

Requirements	Details
User accounts	- Login (as administrator) - Edit user account (permissions, password, …) - Remove user account - Create account - Block user
Server communication	- Connect to server (local / remote) - Transmit queries - Receive/process results - Upload music files (with title/artist and other information)
Manage database	- Create a new songs DB - Drop existent DB - Edit DB information - Remove song from DB / server - Add song to DB / server (and order feature extraction and classification) - Add song annotations (AV values, segmentation info, etc., for validation tests) - Add song information
Manage settings	- View existent audio features (grouped by category) - Edit list of selected audio features - Change distance algorithm used (similarity) - Change default classifier (SVM, GMM, k-NN) - Edit classifier parameters - Change default taxonomy - Perform analysis of results (model accuracy, etc.) - Close users registration

Table 18: Client application requirements (Administration Application)

Appendix B – Software Design

Software Design is the key phase of a software engineering project, it precedes the development phase and both are intimately connected. In this chapter the technical aspects of the application are planned and discussed. From the requirements analysis results, use case diagrams were produced, giving us a visual interpretation of the ideas. Following, some wireframes and prototypes were constructed and more will be done, serving as the main reference for the final application design. Other important aspects belong to data storage. Here, it is important to carefully adopt the right solution for application needs. Even more crucial is to correctly design the data model, ensuring it is robust, stable, with guarantees of good performance and scalability, handling big amounts of data and application redesigns.

These last steps have already been started but in some cases results are very preliminary and will only be concluded in the next month.

Use Cases

The use cases are a type of Unified Model Language (UML) diagrams for graphical representation of system behavior. Use cases describe the interaction between one or more actors and the system itself, represented as a sequence of simple steps, in other words they describe the system from the user's point of view.

During the planning phase a few use case diagrams were produced to illustrate the main actions of the mood tracking application. These use cases will be improved in the future, adding further detail to the requirements.

Next, two use case diagrams will be presented. The first shows how a normal user interacts with the system, detailing the actions he can perform (Figure 37). The second use case (Figure 38) represents the actions an administrator is able to do by using the system (specifically the administration application).

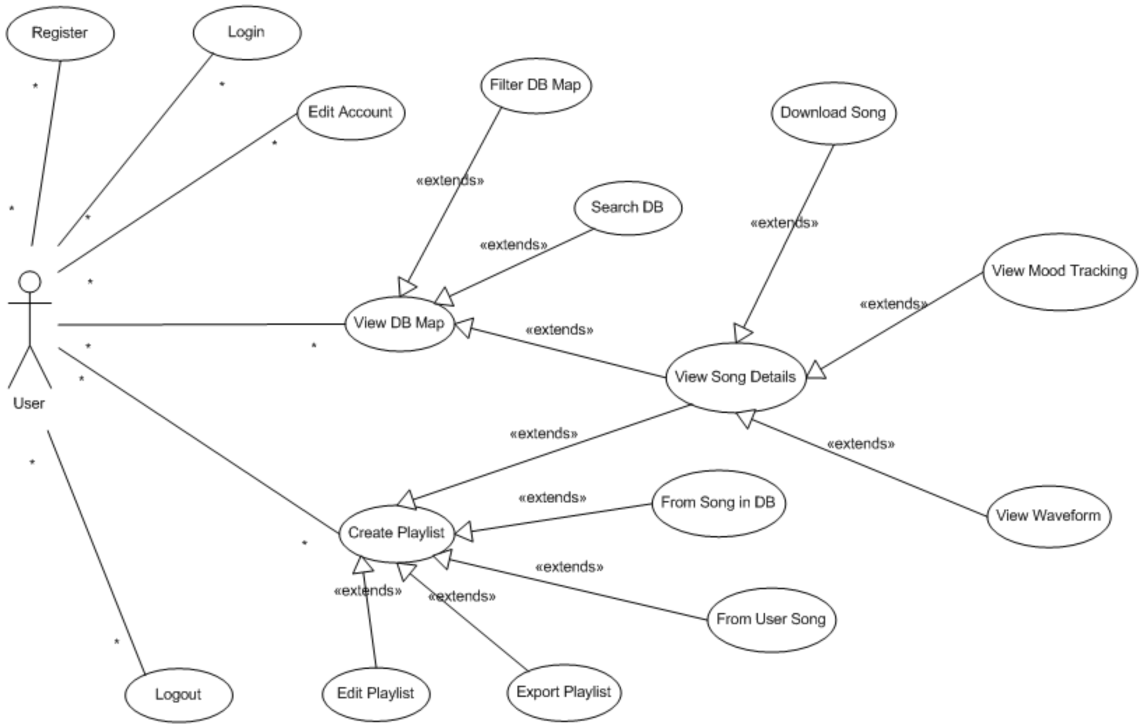


Figure 37: Use case 1 (normal user)

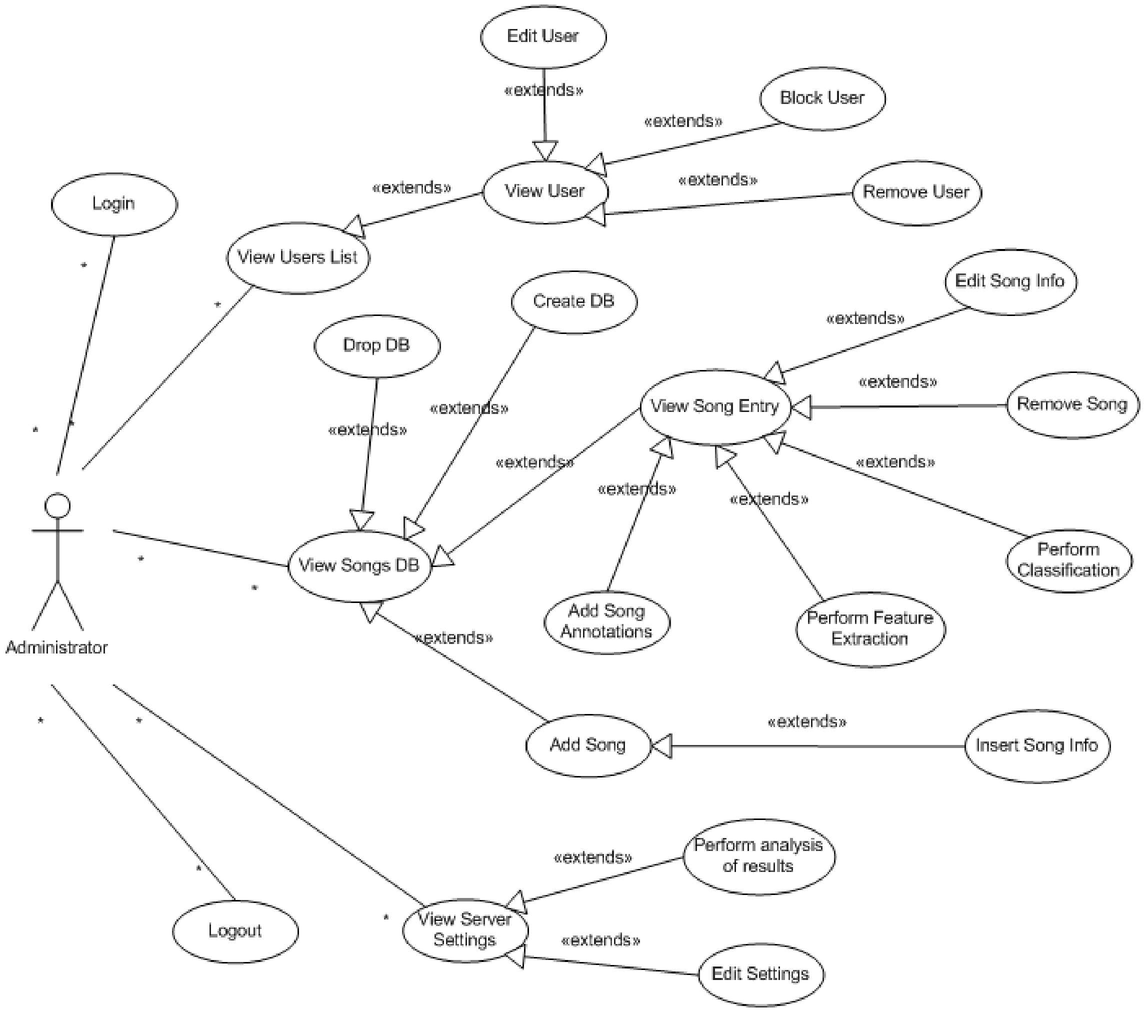


Figure 38: Use case (administrator)

System Architecture

The diagram below (Figure 39) describes the system architecture for our project, following the application requirements outlined in Chapter 3. There, the three applications are represented, as well as the interaction between them, with some of their most important internal modules detailed.

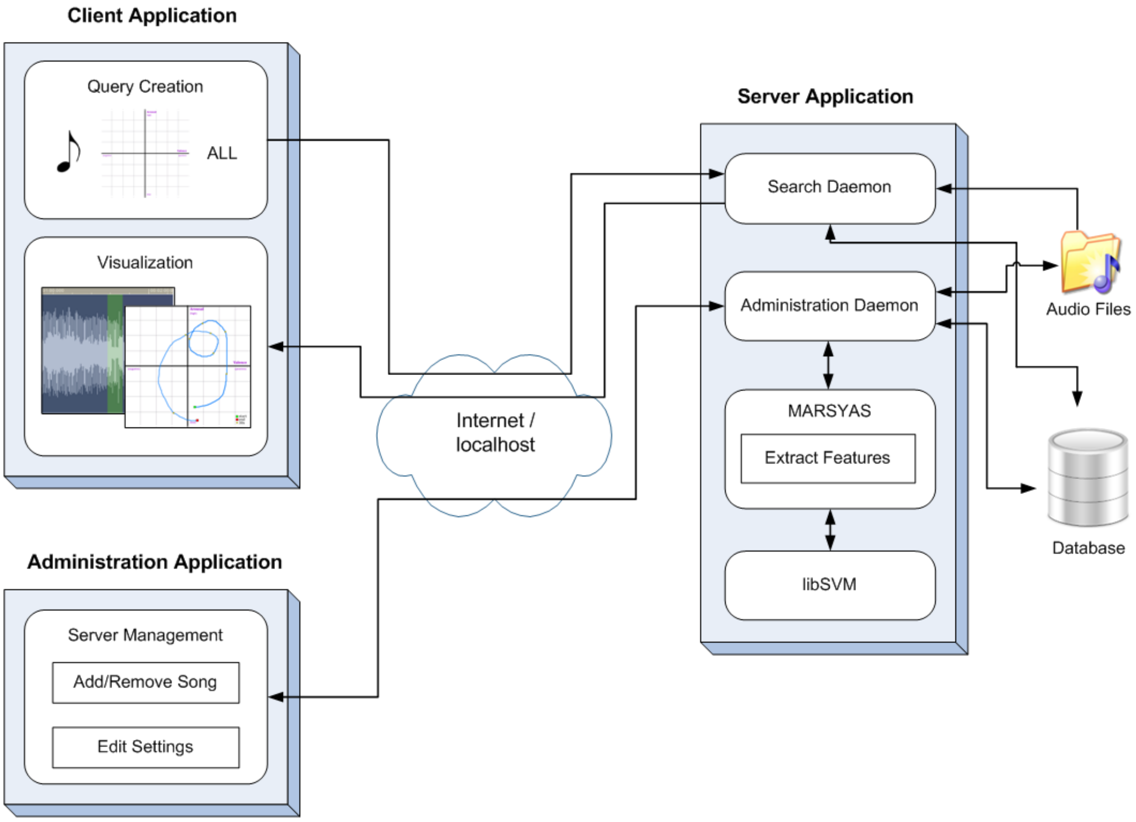


Figure 39: System Architecture

Application Prototype

A prototype is a concept of the future system. As the word implies, a prototype means something in the primitive form, a first impression and thus the images presented below do not represent the final product but the original, primitive and raw ideas for design.

These prototypes are split in different groups, a few small wireframes demonstrating the existent ideas to represent mood and mood tracking information for a specific song, and a more detailed application prototype, whose function is to give a proper representation on how the application will look like and how the GUI is designed.

Mood Tracking Prototypes

So far, two different methods for displaying mood tracking results were chosen and will be presented in the following paragraphs. However, these two are not the only possibilities since new innovative ideas can always surface and be added in the future.

The first method consists of drawing the music mood changes in a 2D graph representing the Thayer's plan. This can be performed in real time, while the music is being listened or at once, retrieving the information directly from the server and showing the complete trace. This idea is described in a sequence of figures presented below (Figure 40, Figure 41 and Figure 42), showing the trace evolution, while crossing all the four quadrants during the song.

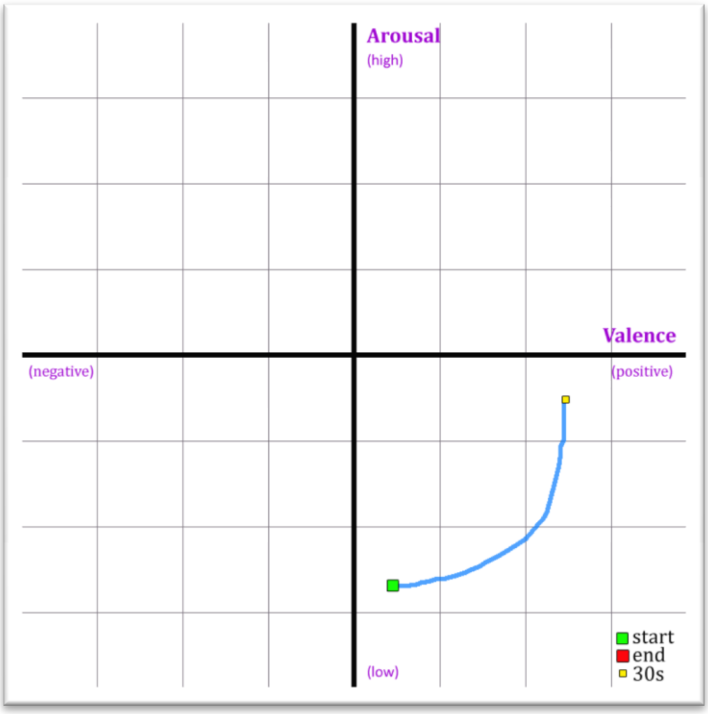


Figure 40: Mood tracking (part 1)

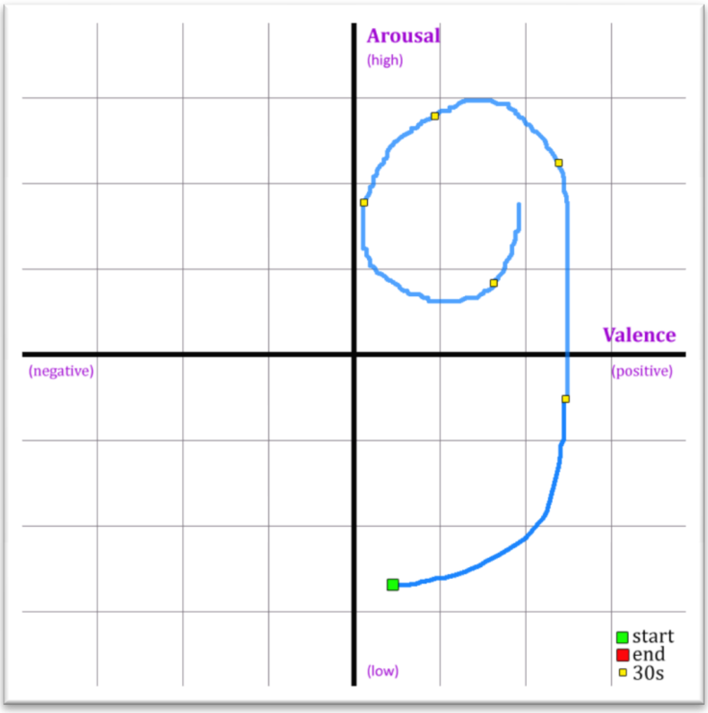


Figure 41: Mood tracking (part 2)

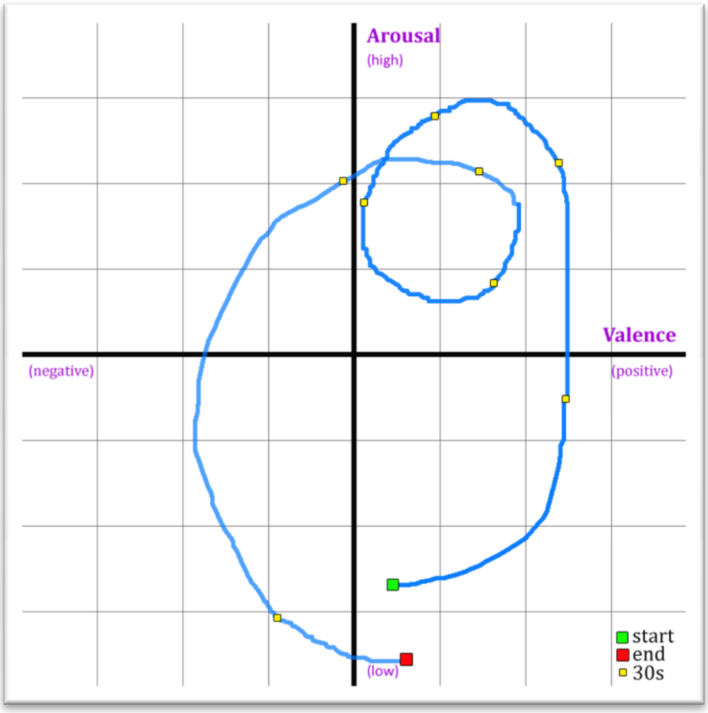


Figure 42: Mood tracking (part 3)

The second method is a bit more technical. This time mood changes are reported using the audio waveform. This view is less detailed than the previous method, only indicating when mood changes between quadrants. Four colors highlight different parts of the signal, representing the four distinct quadrants of Thayer's model of mood (see Figure 43). A possible improvement to this would be to use color gradients or saturation / luminance, marking this way how far from the center the signal is and the proximity with other quadrants. Although this idea may sound useful, it is probably complex to implement and depends on the functionalities of the GUI, being listed as something for future reference, to be done if time and technology permits.

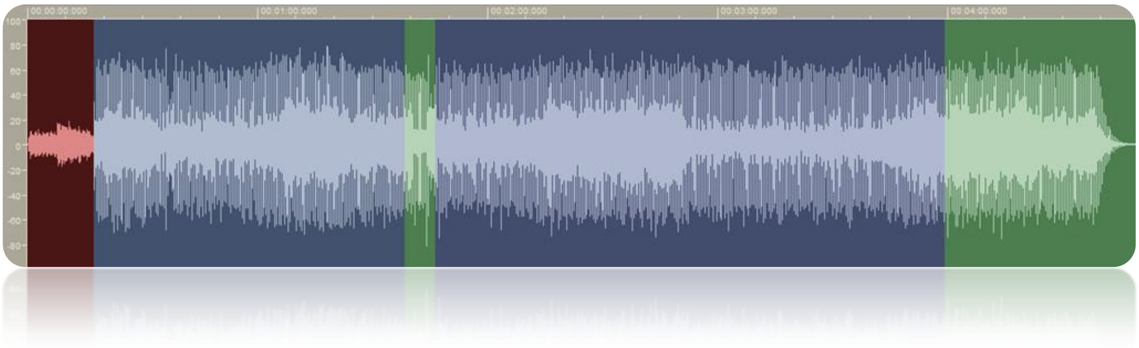
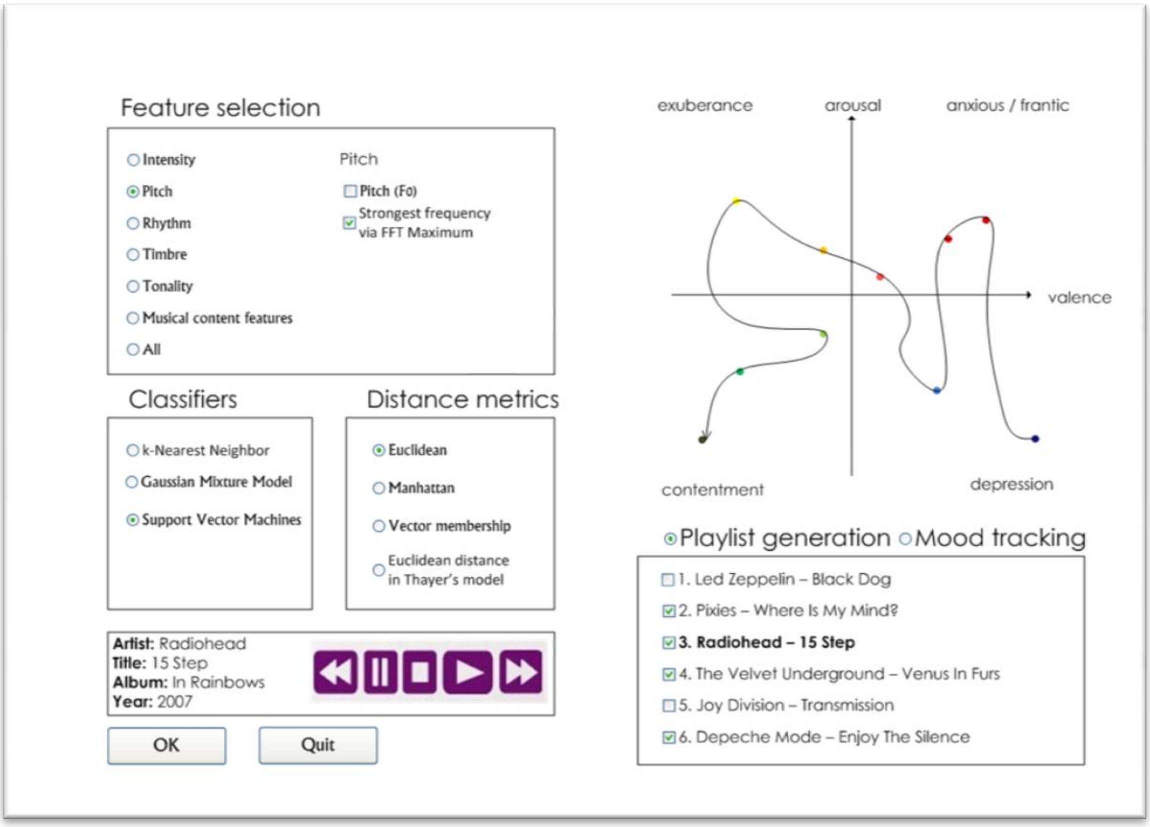


Figure 43: Mood tracking with wave form view

Graphical User Interface Prototype

The GUI will be implemented using the Qt framework. Currently, only small wireframes and prototypes were prepared, documenting in general aspects what we consider essential to be presented in the future interface. In the next picture we introduce the key concepts identified until know (Figure 44).

A 2D graph representing Thayer's model of mood is essential to show a DB map, make searches and filters as well as view mood tracking for a selected song. The commands will be used to play and control a downloaded song, viewing mood in real time and so on. Settings selection like features, classifier and distance metrics are useful but a bit more advanced, with some more related to João's work. From this concept there are other parts missing like wave form visualization, login menus, filter details, song submission and others. These will appear in the next concepts or GUI.



Feature selection

○ Intensity
● Pitch
○ Rhythm
○ Timbre
○ Tonality
○ Musical content features
○ All

Pitch

□ Pitch (F0)
☑ Strongest frequency via FFT Maximum

Classifiers

○ k-Nearest Neighbor
○ Gaussian Mixture Model
● Support Vector Machines

Distance metrics

● Euclidean
○ Manhattan
○ Vector membership
○ Euclidean distance in Thayer's model

Artist: Radiohead
Title: 15 Step
Album: In Rainbows
Year: 2007

● Playlist generation ○ Mood tracking

□ 1. Led Zeppelin – Black Dog
☑ 2. Pixies – Where Is My Mind?
☑ **3. Radiohead – 15 Step**
☑ 4. The Velvet Underground – Venus In Furs
□ 5. Joy Division – Transmission
☑ 6. Depeche Mode – Enjoy The Silence

Figure 44: Client application GUI concept

Data Storage

Choosing the right data storage solution is really important in order to ensure the optimal application performance and operation. During the previous months various alternatives were discussed, from plain text, to XML files and real databases among others. The process of choosing the most adequate one was based on having a clear picture of current and future needs for data storage and access in the project. An acceptable songs library could easily reach the thousands of files. Storing general song information but also mood tracking information (for each second) will generate a significant amount of data. Using simple text or XML files would make it harder and slower in basic operations like search, edits and deletes. On the other hand using a full database management system solution like MySQL, PostgreSQL or Oracle is really exaggerated for our current needs, increasing the hardware requirements as well as the need to have those tools installed and always running.

The need for an application that can be used both in local and remote applications was the key factor on excluding a complete database management system like MySQL. If the objective was only for a powerful and robust server to be used on a remove server our choice would be different. Due to these facts as well as all the limitations that could exist in a local machine and the need for something easy to set up and use, we opted for a lighter solution, making it possible to access all functionalities of the system by running both apps and having the DB file.

The solution chosen is SQLite[22]. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. The database file format is cross-platform, storing all information in a single, small footprint file.

Data Model

Considering our data storage choice (SQLite), the next step is designing how the data will be kept in the database files. The result of this process will be an Entity-Relationship (ER) diagram, an abstract and conceptual representation of data. This part is extremely important to ensure that the information is stored using the minimum space possible and retrieved with great performance. To guarantee it, we will normalize the DB at least until the 3rd normal form and if possible Boyce-Codd normal form (at least for tables we expect to be the most used / biggest).

The DB should keep all information about songs (id, title, artist, file name and others), mood information (at least AV values), users and other needs. Stable versions of the ER and of the physical diagram are presented below (Figure 45 and Figure 46). It already covers most of the needs and avoids duplication of values and null cells, some improvements might be needed in the next months.

[22] http://www.sqlite.org/

Entity-Relationship Model Diagram (mood.db)

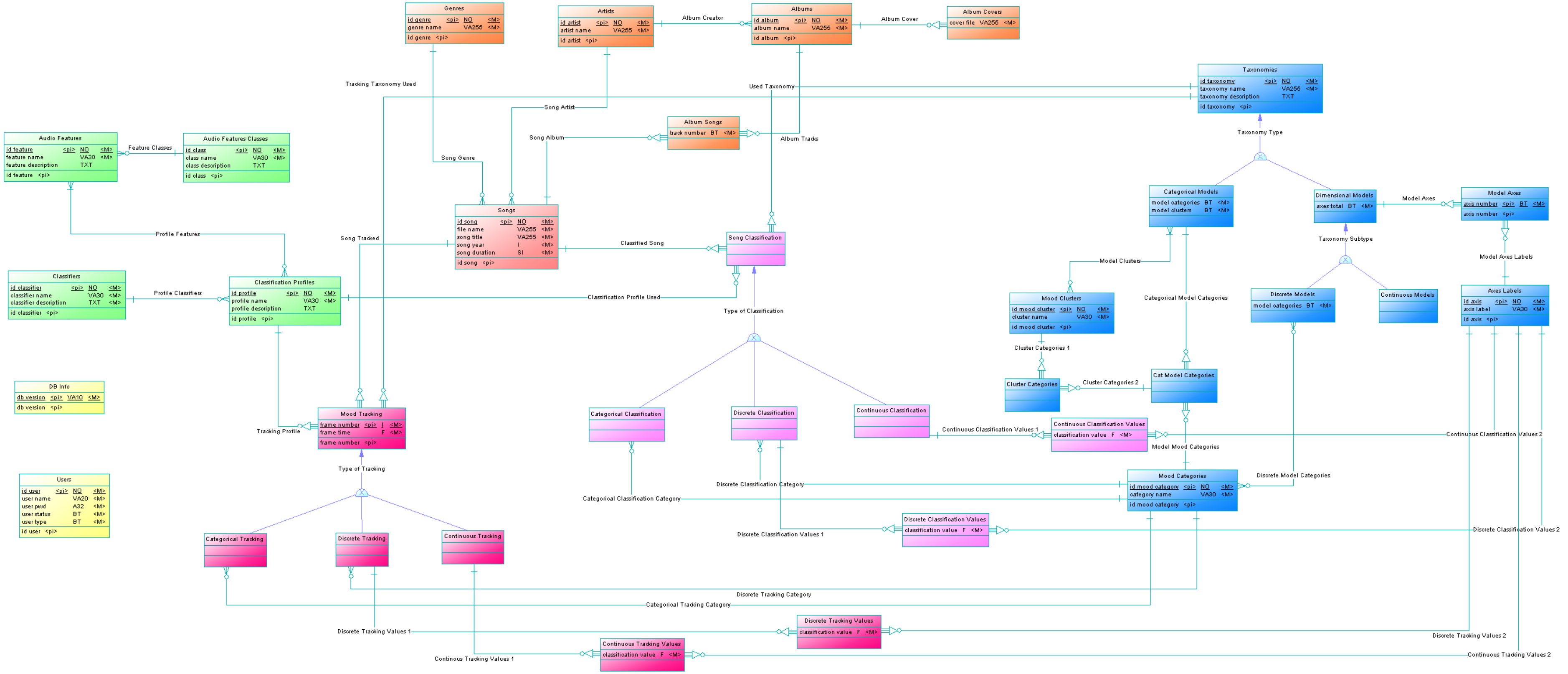


Figure 45: Entity-Relationship model diagram

Physical Data Model Diagram (mood.db)

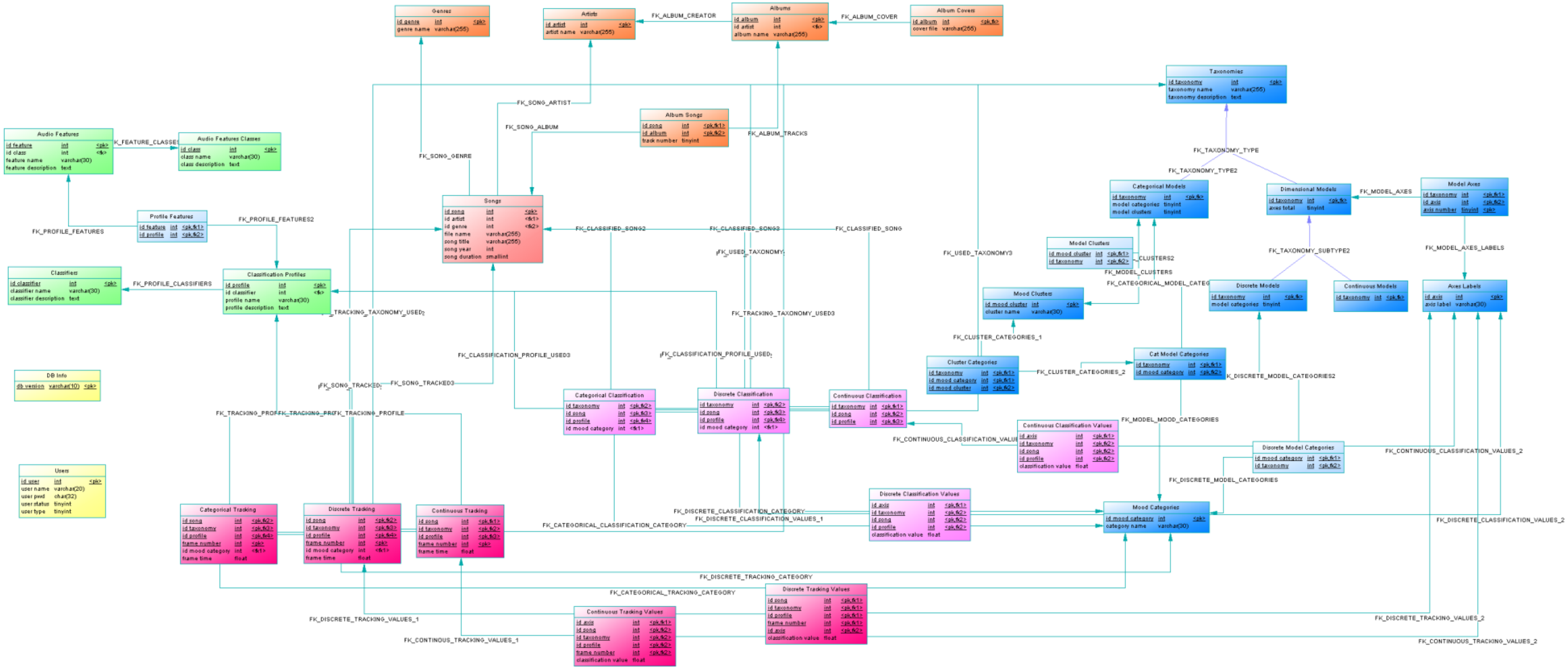


Figure 46: Physical Data Model diagram

Appendix C – Marsyas Overview

Marsyas is a software framework, written in C++, designed for audio analysis and synthesis.

The basic idea behind the design of Marsyas is that any audio analysis/synthesis computation can be expressed as some type of processing objects called *MarSystems*. Networks of these blocks can also be combined and encapsulated as one single *MarSystem*.

A good number of these building blocks that form the basis of many published algorithms are already provided, used to implement more complicated algorithms. In addition, it is also possible to extend the framework by creating new blocks. Dataflow in the framework is synchronous and each time the tick() function is called a data slice is propagated across the entire dataflow network. Therefore, a tick can be seen as an instant of time.

MarSystems usually have controls, where they store additional information that may be needed to their correct behavior. One example of this is the *SoundFileSource MarSystem*, which obviously needs the name of the file to be opened. In it there are also other controls like *Gain*, which can be adjusted at any time. To access data blocks inside a network, the framework uses a path notation. A path like "Series/playbacknet/Gain/g1/real/gain" is the control name for accessing the gain control of a *Gain MarSystem* named g1 in a *Series* composite named playbacknet. The framework also provides a mechanism for linking top-level controls to the longer full path control names.

As stated, there are innumerous *MarSystems* available, each one with a different function. Therefore, the most relevant ones will be described below.

Series

This is the most basic structure for connecting *MarSystems* (in Series, as the name implies) into dataflow networks. The output of the first object existent inside the Series will become the input of the second and so on, as shown in Figure 47.

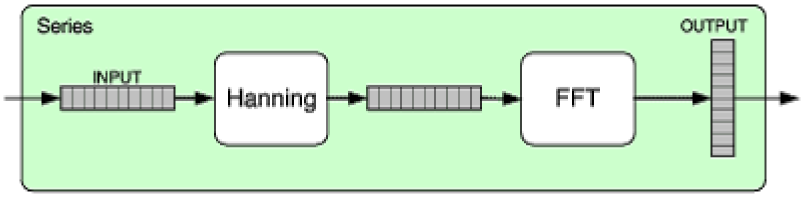


Figure 47: Series composite MarSystem

Parallel

The *Parallel* composite is used to do parallel calculations when there is an input with multiple observations. It receives the input (i.e. various channels) and sends each observation to a different *MarSystem* where calculations run in parallel (Figure 48).

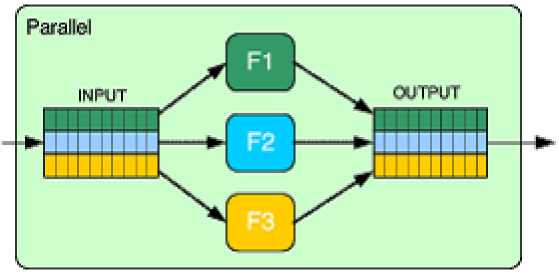


Figure 48: Parallel composite MarSystem

Fanout

Fanout is similar to *Parallel*, but takes a single observation and sends a copy of this observation to all the *MarSystems* inside of it as shown in Figure 49.

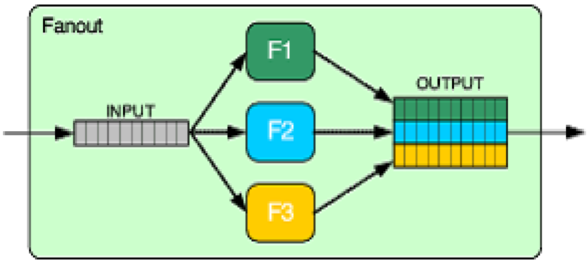


Figure 49: Fanout composite MarSystem

Accumulator

The *Accumulator*, as the name implies, accumulates results of multiple tick process calls to the internal *MarSystem*. If its *nTimes* control is set to 10, for each tick received by Accumulator, 10 ticks will be sent to its internal *Marsystems* (Figure 50).

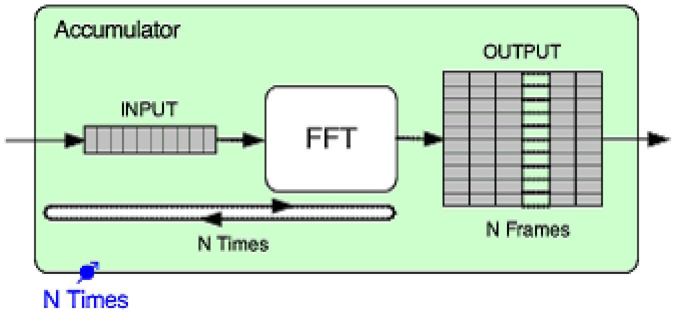


Figure 50: Accumulator composite MarSystem

Shredder

A *Shredder* composite *MarSystem* does the inverse to what an *Accumulator* does. While an *Accumulator* builds up a vector containing the results from multiple ticks, a *Shredder* splits this vector into multiple chunks, effectively increasing the rate at which data is output.