Imperial College London

Department of Computing

Turning Music Into Game

Author:
Paulina Koch

Supervisor:
Dr. Iain Phillips

2nd Marker:
Dr. Robert Chatley

June 2015

Chapter 1

Abstract

Music games present a highly pervasive new platform to create, perform and appreciate music. In this project we will attempt creating a music rhythm game which, given a music track, extracts its features to generate a level without human intervention.

This report details the design of such a program and evaluates its effectiveness. The development of the program has lead to the discovery of new and powerful algorithms in music analysis, as well as successfully demonstrating the power of computers in developing creative works.

Chapter 2

Acknowledgments

I would like to thank my supervisor, Iain Phillips, for his sharp insight into the problems encountered, and his uncanny ability to immediately suggest a good solution for each one.

Contents

Chapter 3

Introduction

> Music and games share a fundamental property: both are playable, offering their listeners and operators an expressive experience with the framework of melody and rhythm [1].

As the quote suggests, both games and music have one thing in common — the act of playing. Just as player's character might die in an attempt to complete a level, causing him to lose the game, the pianist can fail at the attempt of performing a musical piece.

Perhaps this analogy inspired programmers to develop a new genre of games - music games. Music games are games in which players interact with music. Possibly the most commonly known franchises in this genre are Guitar Hero, Rock Band and Dance Dance Revolution. In this type of games user has to follow the indicators on the screen telling him which buttons to hit.

The concept of a music game stormed the industry in 2005, after Guitar Hero was released. The project soon turned into the fastest new video game franchise to reach $1 billion in retail sales in the history of the business, with Guitar Hero III being the first game to reach $1 billion [2].

However, a limited amount of songs transcribed and adjusted to the game play soon caused the popularity of such music video games to decline. Some brave fans of the franchises took it upon themselves to transcribe songs to create new levels. The producers, seeing the tendency, started releasing the in-app purchases to enable the players to extend their library and thus, keep the users.

Due to the time consuming and difficult nature of the process of manually adding new songs, most players usually limit themselves to pre-processed songs provided by the game producers, not really taking advantage of the full capabilities of the games.

This project aims to change the way users look at the music rhythm games. We are creating a game which will allow them to upload any song they would like and automatically generate a Guitar Hero-like level corresponding to it.

This will be achieved by implementation of a melody extraction from polyphonic music signals algorithm using pitch contour characterisation. The algorithm consists of four parts - sinusoid extraction, salience function, pitch contour creation and melody selection. In this approach, pitch contours - time continuous sequences of pitch candidates, are grouped using auditory streaming cues. To filter them, we define a set of contour characteristics, which help distinguish between melodic and non-melodic contours. This leads to the development of new voicing detection, octave error minimisation and melody selection techniques [3].

We will then design and develop an algorithm for mapping the extracted to a series of buttons on the screen to create an interesting and challenging game for a user, as no literature describing such problem was found so far.

In addition to this, we will attempt to develop a mood extraction algorithm to dynamically generate surroundings in the game. Specifically, we treat music emotion recognition as a regression problem to predict the arousal and valence values (AV values) of each music sample directly, which then can be used to generate unique surroundings for every level generated. This continuous view of music emotion makes the proposed music emotion recognition system free of the inherent ambiguity issue. In addition to this, because there is more freedom in describing a song compared to defining and assigning mood classes, the subjectivity issue is alleviated to some extent. [4].

The music emotion recognition will be achieved by designing and training a neural network to predict listeners' mean valence and arousal ratings associated with musical pieces.

With this project we would also like to show that sophisticated academic music analysis techniques can be combined together and applied to real world problems in an efficient and reliable manner.

Finally the project aims to be more than just a research study of feasibility. The result of successful completion will be an application of sufficient reliability and quality that it can be released to, and used by, untrained computer users. To our knowledge, it is the only computer game allowing people to generate Guitar Hero-like levels that also generates the surroundings tailored to every music track.

Chapter 4

Background

In this section, we investigate different types of music games [5], along with a deeper look into Guitar Hero, on which we base our main concept for the gameplay. This is followed by a discussion of the most applicable publications in music analysis, on finding the main melody in a musical track in particular.

4.1 Music Video Games

A music video game can be defined as a type of game that uses music or rhythm as an integral part of gameplay. This may involve pressing buttons in time with a song, whether on a conventional controller, and instrument controller or some kind of dance mat, singing into a microphone or creating original music. Players can often perform different parts of the same song together in local multiplayer games or over the Internet, providing enjoyable social experiences [7].

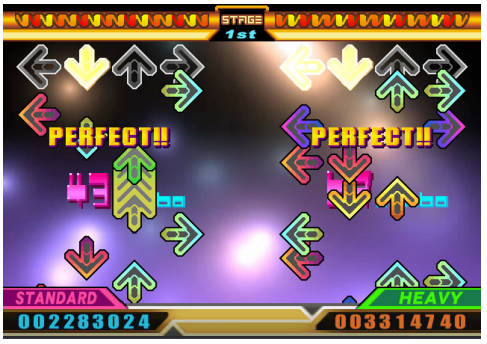


FIGURE 4.1: Screenshot from Dance Dance Revolution, an example of a rhythm music game [6].

Some games exhibit a sandbox style that encourages a free-form gameplay approach whereas other a hybrid style, which combines musical elements with more traditional genres, for example puzzle games or shooters.

Below we will briefly go over different types of music video games that can be found on the market.

(A) is - Internal Section - an example of a generative hybrid music game [8].

(B) SimTunes - an example of a free form music game [9].

FIGURE 4.2: Examples of music video games.

4.1.1 Music Memory Games

The goal of the music memory game is to score a player on their musical memory. Music track is presented to the user who then has to provide an appropriate response to each prompt from the game. Games may be based on different primary musical aspect (whether it is the rhythm, pitch or volume). However, a vast majority of the releases available on the market are rhythm-based.

Rhythm games typically focus on dance or the simulated performance of musical instruments, and require players to press buttons in a sequence dictated on the screen. Doing so causes the game's protagonist or avatar to dance or to play their instrument correctly, which increases the player's score [10]. An example of such games could be Guitar Hero or Dance Dance Revolution.

4.1.2 Hybrid Music Games

Hybrid music games are characterised by substantial and meaningful interactions between a player and the music game in a game that apparently belongs to a non-musical genre. This type of games can be further split into two sub-types.

Generative music video games make use of user's actions. By monitoring interaction with the surroundings in the game, the mechanism generates sounds that are then integrated into the soundtrack, permitting the player's direct interaction with the score. This encourages the creation of a synesthetic experience — when upon stimulation of one sense others activate, causing an involuntary experience. An example of such game could be Rez, which is a simple rail shooter. However, thanks to integrating sounds generated by player completing the normal task of rail-shooting, the musical score is dynamic.

(A) Screenshot from Guitar Hero - player is attempting to play a song [11].

(B) A guitar shaped controller used in the game [12].

FIGURE 4.3: Guitar Hero components

Reactive music games, in contrast to generative one, employ music to determine the gameplay. In such games, the player takes cues from soundtrack to devise his gameplay. For example, iS - internal section, uses the music to determine the dynamics of the non-musical components of the game.

4.1.3 Free Form Music Games

In free form music games, the main task of the user is to create content. This form of music game is often compared to non-game music synthesisers. Free form music games are somewhere between generative hybrid music games and non-game utilities, depending on the degree to which their gameplay relies on a driving underlying plot-line. An example of such game could be SimTunes, where the user is painting a picture using large pixels and each colour represents a musical note.

4.2 Case Study - Guitar Hero

Guitar Hero is one of the most popular franchises in the history of music games. The first of the series was published in 2005 by RedOctane and Harmonix. In the games, players instrument-shaped game controllers to simulate playing the instruments across numerous rock music songs. It is widely considered a highly entertaining game fully embracing the rhythm-based music game.

4.2.1 The Controller

Rather than a typical gamepad, Guitar Hero uses an instrument-shaped controller (guitar in the earlier releases, bass, microphone and drums in more recent ones). Playing the game with the guitar controller simulates playing an actual guitar, except it uses five

coloured “fret buttons” and a “strum bar” instead of frets and strings, and an analogous mapping for the other instruments. They incorporate most of the real life techniques and motions that an instrumentalist would perform on a real instrument.

4.2.2 The Gameplay

The actual game itself works exactly as many other music titles do. At the bottom of the screen, a number of (varying depending of level of difficulty) buttons is shown. In each attempt, a series of notes moves across the screen and when a note aligns with a button, player is supposed to press a corresponding button, gaining points depending on the accuracy. If the player failed to achieve a certain amount of notes — his performance meter stays low for a longer time, he loses the game.

However, there are a couple minor improvements that Harmonix has made to the general music game formula. By pressing buttons with really good accuracy in a song, a player is able to build up Star Power, which when unleashed, doubles up current point multiplier. Star Power also adds a bit of a strategic element - player not only earns more points when it is activated, but he can also raise your performance meter faster, enabling him to last longer when encountering a trickier part of a song.

4.2.3 The Critique

Without a doubt, Guitar Hero features a great selection of music. However, there will always be tracks missing, regardless of how many versions of Guitar Hero are released. People have different tastes and limiting a game to a set of tracks that everybody is supposed to enjoy is a really hard task.

Some more advanced users familiar with Computer Science attempted to transcribe songs and to create new levels. However, this process was really difficult, consisting of many laborious stages and requiring an additional midi files with separated guitar track. This discouraged an average user from fully making use of game’s capabilities. The producers, seeing the tendency, started releasing the in-app purchases to enable the players to extend their library and thus, keep the users.

As there is a clear need for custom music extension to the game, implementing a feature of uploading some music preferred by the player would definitely improve user satisfaction. However, this has not been achieved yet as the task itself is quite complex. Moreover, enabling the users to load in some music would deprive the company of their income sources.

4.3 Introduction to Music Analysis

Automatic music analysis is the automated extraction of relevant perceptual information (notes, instruments, etc.) from music files (like mp3s). First attempted in the 1970s at Stanford University [Moorer], it remains an unsolved problem. The problem is highly multifaceted and interdisciplinary, requiring the extraction of musical notes, instruments, percussion, emotion, etc., and drawing from fields as varied as computer science, mathematics, biology, physics, psychology, and electrical engineering. The problem's difficulty lies in a necessity to reverse-engineer the human brain.

For a long time people were researching ways of estimating the fundamental frequency, be it with monophonic music recording or multi-pitch estimation. Melody extraction differs from both of those problems — unlike monophonic pitch estimation it handles polyphonic tracks and in contrast to multi-pitch estimation, it must also include a mechanism for source identification, to spot the voice carrying the melody within the polyphony. To be able to evaluate the performance of the new algorithms, annual Music Information Retrieval Evaluation eXchange (MIREX) has been running since 2005. In this campaign, different models are evaluated against the same sets of music collections in order to obtain a quantitative comparison between methods and assess the accuracy of the current state-of-the-art in melody extraction [13].

4.3.1 Sound Spectrum

Most sounds are made up of a complicated mixture of vibrations. A sound spectrum is a representation of a sound – usually a short sample of a sound – in terms of the amount of vibration at each individual frequency. It is usually presented as a graph of either power or pressure as a function of frequency. The power or pressure is usually measured in decibels and the frequency is measured in vibrations per second (or hertz, abbreviation Hz) or thousands of vibrations per second (kilohertz, abbreviation kHz).

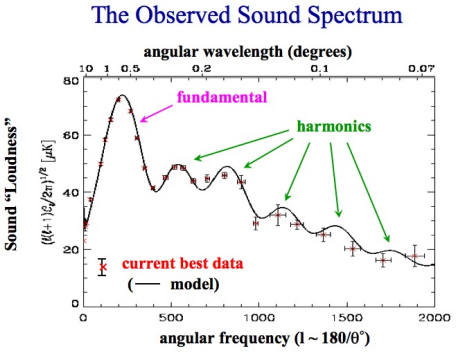


FIGURE 4.4: Example of a sound spectrum diagram [14].

4.3.2 Pitch, Tones, Fundamental Frequency

Pitch is the most natural way of ordering sounds on a frequency-related scale. If sounds whose frequency is clear and stable enough to be distinguished from noise, they can be compared among one another as "lower" or "higher". Pitch is not an objective physical

property — it depends on anatomy and physiology of the auditory system, which is a subject of an extensive study called psychoacoustics.

A semitone is the smallest musical interval commonly used in Western tonal music. Two semitones constitute a tone.

The fundamental frequency f_0 is defined as the lowest frequency of a periodic waveform. A harmonic (or a harmonic partial) is any of a set of partials that are whole number multiples of a common fundamental frequency. This set includes f_0, which is a whole number multiple of itself (1 times itself).

Fundamental frequency can be thought of as the physical property most closely related to perception of pitch. This is why in this context pitch and fundamental frequency can be used interchangeably.

4.3.3 Polyphonic Music

Polyphony is a word derived from Greek poluphōnosis meaning more than one sound — a texture consisting of two or more simultaneous lines of independent melody. This can be contrasted with homophony, where musical parts move generally in the same rhythm and one dominant melodic voice is accompanied by chords or monophony, where only one voice is found.

However, in our case, the term polyphonic will simply refer to any type of music in which two or more notes can be played simultaneously. This can be achieved either by playing in different instruments (for example, voice, guitar and bass) or a single instrument capable of playing more than one more at a time (like a piano).

4.3.4 Melody

The concept of "melody" ultimately relies on the judgement of people listening. This is why it will vary depending on the application context - whether we want to determine symbolic melodic similarity or transcribe a music track.

In order to have a clear framework to work within, the Music Information Retrieval (MIR) community has adopted in recent years the definition proposed by [15], "...the melody is the single (monophonic) pitch sequence that a listener might reproduce if asked to whistle or hum a piece of polyphonic music, and that a listener would recognise as being the 'essence' of that music when heard in comparison".

In practice, research has focused on "single source predominant fundamental frequency estimation" — which means a search for a main melody coming from a single sound source throughout the song analysed. As we can see, the subjective element is still present in this description of a melody as there might not be a definite way of deciding

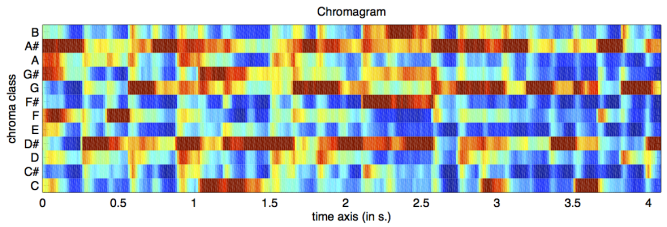


FIGURE 4.5: Example of a chromagram.

what predominant is. However, it fits well with our project's objective — generating a game level based on changes in the pitch.

4.3.5 Filter

Any medium through which the music signal passes, whatever its form, can be regarded as a filter. However, we do not usually think of something as a filter unless it can modify the sound in some way.

A digital filter is a filter that operates on digital signals, such as sound represented inside a computer. It is a computation which takes one sequence of numbers (the input signal) and produces a new sequence of numbers (the filtered output signal) [16].

4.3.6 Short Time Fourier Transform

Short-time Fourier transform (STFT), is a signal processing method which is used in analysis of non-stationary signals with statistic characteristics varying with time. In particular, STFT extracts several frames of the signal to be analysed with a window that moves with time. If we set the window size to be narrow enough, each frame extracted can be viewed as stationary so that Fourier transform can be used. With the window moving along the time axis, the relation between the variance of frequency and time can be identified [17].

The short time Fourier transform of a time-domain signal y is denoted by the matrix $F \times N$, F being the Fourier transform size and N the number of analysis frames.

4.3.7 Chroma

A chroma feature is characterised by a 12-dimensional vector representing the amount of energy that can be found in each of different pitches. Usually 12 pitches commonly existing in the western popular music folded into one single octave are considered, but one can use multiples of 12 to consider semi-tones or even smaller differences between tones. This is achieved by applying a constant-Q transform across the entire spectrogram.

In mathematics and signal processing, the Constant Q Transform transforms a data series to the frequency domain. It is related to the Fourier Transform,[1] and very closely related to the complex Morlet wavelet transform.[2]

The transform can be thought of as a series of logarithmically spaced filters, with the k-th filter having a spectral width some multiple of the previous filter's width, i.e.

$$\delta f_k = 2^{\frac{1}{n}} * \delta f_{k-1} = \left(2^{\frac{1}{n}}\right)^k * \delta f_{\min} \tag{4.1}$$

where δfk is the bandwidth of the kth filter, fmin is the centre frequency of the lowest filter, and n is the number of filters per octave.

Next, the spectrogram is folded into one octave comprising the M quantised pitches, where M is the number of pitches found in the chroma. When these features are stack together following the song structure in a N x M; matrix, a chromagram is generated, where N is the number of time frames in which the musical piece has been divided.

4.4 Main Melody Extraction from Polyphonic Music

In this section we will go over two different approaches to the problem of main melody extraction from polyphonic music, using source separation and a salience function. Then we will compare both methods to determine which one is more suitable for our project.

4.4.1 Source Separation Based Approach

In polyphonic tracks the main melody can be represented by a specific source/filter model. In case of the leading vocal part, the vocal cords are treated as a source and the voice tract as a linear acoustic filter.

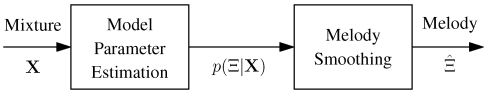
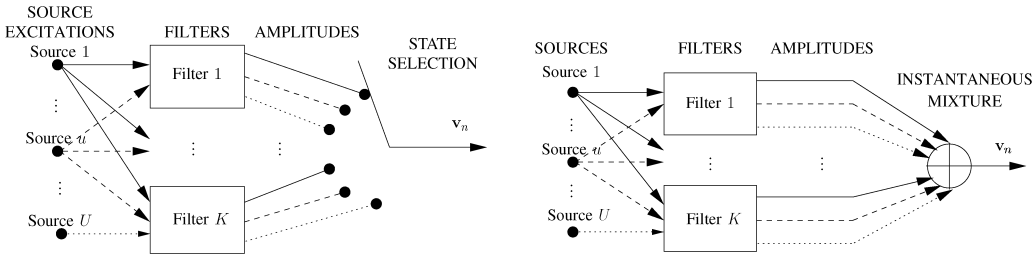


FIGURE 4.6: Outline of system proposed by Durrieu: X is the STFT of the mixture signal, $p(\Xi|X)$ the posterior probability of a given melody sequence, and $\hat{\Xi}$ the desired smooth melody sequence [18].

In their paper from 2011 [18], authors presented an algorithm in which they assume that at any given time the signal observed is a mixture of two elementary signals - one corresponding to the main source and one to the background music. Therefore, the signal can be represented in an equation $x(t) = v(t) + m(t)$, where $v(t)$ stands for the source of the main melody and $m(t)$ is the background music. Interestingly, this equation also holds for the short time Fourier transform (STFT) X, V and M respectively: $X = V + M$. The models proposed by Durrieu essentially aim at constraining the shapes of these STFT using temporal and spectral constraints.



(A) Schematic principle of the generative GSMM. Each source u is filtered by each filter k. For frame n, the signal is then multiplied by a given amplitude and a "state selector" then chooses the active state.

(B) Schematic principle of the generative IMM. At each frame, all the U sources, each filtered by K filters, are multiplied by amplitudes and added together to produce the leading voice signal.

FIGURE 4.7: Diagram of both models presented in the paper [18].

The likelihood of the vocal part V is calculated using two different frameworks.

The first submission uses the source/filter Gaussian scaled mixture model (GSMM). In this model the source element refers to the excitation of the vocal folds and is therefore linked to the fundamental frequency of the sound f_0, while the filter part is characteristic of the vocal tract shape. This space of possibilities is then discretised so that we consider one possible filter frequency response, which is then used to calculate the likelihood of the vocal part knowing the filter and f_0.

Figure 4.7a A) shows the diagram of the GSMM model for the main voice part. Each source excitation u is filtered by each filter k. The amplitudes for a frame n and for all the couples (k, u) are then applied to each of the output signals. At last a "state selector" sets the active state for the given frame.

The second model was derived from the first one to find a solution that would be more efficient to compute. The authors came up with a formulation that keep the source/filter model within an instantaneous mixture framework (IMM). In this model, for each source a set of filters is defined and at each frame, once every source is filtered and multiplied by a given amplitude, they are all added together.

The background music signal $m(t)$ can be thought of as a mixture of R independent Gaussian sources $m_r(t)$. Each of the sources is centred and characterised by its power spectral density (PSD), which describes how the power of a signal or time series is distributed over the different frequencies. PSD can be estimated using a Covariance Method. Due to the linearity of the Fourier transform, $M(f, t)$, the STFT of m, is also the instantaneous mixture of the R spectra $M_r(f, t)$ of the sources: $M_r(f, t)$. This together with STFT and an amplitude coefficient associated with each source is used to calculate the likelihood for each of the frequency bins. Let $M_t(f)$ be the STFT of the background signal at frame t and frequency bin f, then we write its likelihood.

Once the parameters are estimated using the maximum likelihood criterion for each of the model, the Viterbi smoothing of the melody line is applied, obtaining a trade-off between the smoothness of the melody and its global energy in the signal. The Viterbi

algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events [19].

The authors then parametrise the transitions between the possible main melody without disabling jumps from one note to the other. Using Wiener filtering - digital signal processing reducing the noise, using an statistical estimate of the signal using a desired data without such noise, a framework is implemented to separate the source. This way separated signals are obtained. Computing the energy for each frame of the separated main melody and thereafter thresholding allowed to discriminate between spurious notes and true positives.

4.4.2 Salience Based Approaches

This approach has been the most popular so far, with majority of algorithms evaluated at MIREX implementing it. It and can be split into several smaller stages, as seen in Figure 4.8. In particular, a method implemented in paper [3] seems to be quite promising.

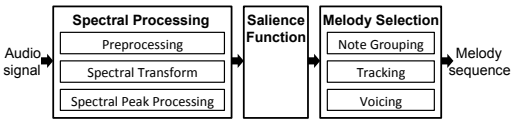


FIGURE 4.8: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creation and melody selection [13].

Usually as a first step, some sort of preprocessing is applied to the audio signal, usually to enhance the frequency content where we expect to find the melody. In particular, Salamon and Gómez apply an equal loudness filter, which enhances the frequencies to which the human ear is more perceptually sensitive, by taking a representative average of the equal loudness curves and filtering the signal by its inverse.

This stage is followed by spectral transform — the signal is chopped into time frames and a transform function is applied to obtain a spectral representation of each frame. This is achieved by applying the Short-Time Fourier Transform given by:

$$X_l(k) = \sum_{n=0}^{M-1} w(n) \times x(n + lH) e^{-j\frac{2\pi}{N}kn} \tag{4.2}$$

with a window length of 46.4ms. Here, $x(n)$ is the time signal, $w(n)$ the windowing function, l the frame number, M the window length, N the FFT length and H the hop size. Thanks to choosing a relatively small hop size, Salamon and Gómez achieve sufficient frequency resolution to identify different notes while maintaining adequate time resolution to track pitch changes in the melody over a short time.

Having done this, we move to frequency/amplitude correction, where the spectral peaks are detected and used to construct a salience function. To avoid a relatively large error in the estimation of the peak frequency caused by binning them in the process of FFT, peak's instantaneous frequency and amplitude are calculated.

As we can see in Figure 4.8, those three steps constitute the spectral processing. But at the core of the salience based algorithms lies the multi-pitch representation, i.e. the salience function — a representation of pitch salience over time. The peaks of this function form the f_0 candidates for the main melody. In the algorithm described by Salamon and Gómez, this computation is based on harmonic summation, where the salience of a given frequency is computed as a sum of the weighted energies found at harmonics (integer multiples) of that frequency. Using only the peaks for the summation allows the authors to discard less reliable values and apply further frequency corrections.

The salience function presented in the paper covers a pitch range of nearly five octaves from 55Hz to 1.76kHz.

Peaks of the salience function at each frame are now potential f_0 of the main melody. At this point some methods for melody extraction attempt to track the melody. However, Salamon and Gómez filter out the non-salient peaks, first by comparing them to the highest peak in the frame and then to a value computed using salience mean and standard deviation of all remaining peaks (in all frames). Now the peaks are grouped into pitch contours - time and pitch continuous sequences of salience peaks as shown in Figure 4.9.

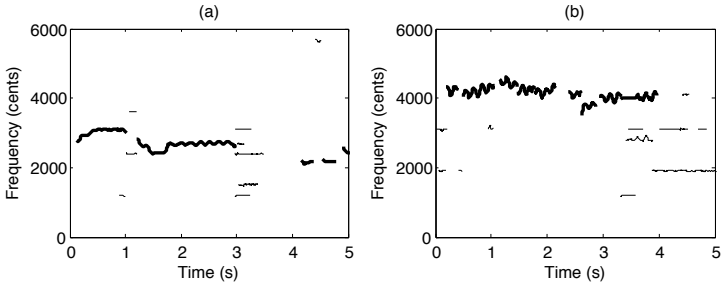


FIGURE 4.9: Pitch contours generated from excerpts of (a) vocal jazz and (b) opera. Melody contours are highlighted in bold [3].

Having created the pitch contours, Salamon and Gómez are faced with the task of determining which one belongs to the main melody. The authors define features based on contour pitch, length and salience.

Given the peaks of the salience function, we now have to determine which pitch values belong to the melody. This process is initiated by grouping peaks into continuous pitch contours, out of which a melody is selected later.

The next main block in this algorithm shown in Figure 4.8 is the melody selection which is comprised of three steps: voicing detection, octave error minimisation/pitch outlier removal, and final melody selection. As the name suggests, the aim of the voicing detection is to determine when the melody is present.

To filter out these contours Salamon and Gómez take advantage of the contour mean salience distribution. By setting the threshold to a value slightly below the average contour mean salience of all contours in the excerpt C_s, we can filter out a considerable amount of non-melody contours. The authors define the following voicing threshold τ_v

based on the distribution mean C_s and its standard deviation $\sigma_{\overline{s}}$:

$$\tau_v = C_s - v \times \sigma_{\overline{s}} \tag{4.3}$$

The parameter v determines the lenience of the filtering - a high v value might keep the false melody contours and a low value might filter out the melody contours.

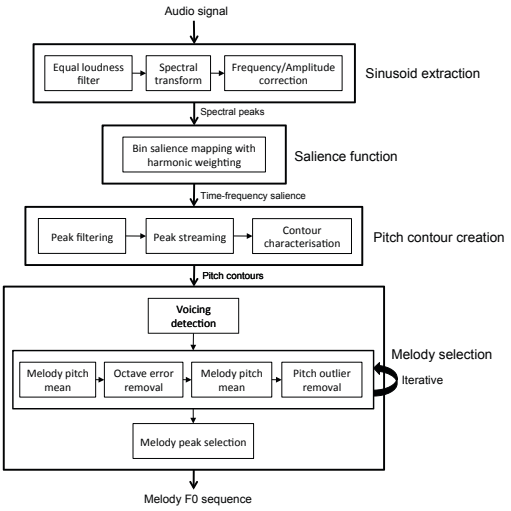


FIGURE 4.10: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creation and melody selection [3].

It is also important to note that detecting certain characteristics in the contour increases a probability of it being the melody contour, for example in case of detecting a vibrato - a regular, pulsating change of pitch, used to add expression to vocal and instrumental music. [20]

Next step in the melody selection described by Salamon and Gómez in their paper is octave errors and pitch outliers removal.

In particular, the octave errors are the main sources of errors in melody extraction systems, when a multiple or submultiple of f_0 is reported as the main melody.

To detect such errors, contour trajectories are compared by computing distance between their values on a per-frame for the region they overlap in and computing the mean over this region. If the mean distance is within 1200 ± 50 cents, the contours are considered octave duplicates.

Secondly, Salamon and Gómez use the relationship between neighbouring contours (in time) to decide which of the duplicates is the correct one. Their approach is based on two assumptions: firstly, that most (though not all) of the time the correct contour will have greater salience than its duplicate (the salience function parameters were optimised to this end). Secondly, that melodies tend to have a continuous pitch trajectory avoiding large jumps, in accordance with voice leading principles.

The method iteratively computes the $\overline{P(t)}$ - pitch trajectory that represents the time evolution of the melody's pitch. It then detects and removes an octave duplicate as well as the "pitch outliers" – contours more than one octave above or below the pitch mean and then it is recalculated. Authors empirically discovered that 2 iterations of this process are enough to get a good approximation of the true trajectory of the melody, which is then passed to the final stage of the model - the final melody selection.

At this stage, there is often only one peak to be chosen as the main melody. When there is still more than one contour present in a frame, the melody is selected as the peak belonging to the contour with the highest total salience $C_{\sum s}$. If no contour is present the frame is regarded as unvoiced.

4.4.3 Comparison of both approaches

In their paper [13], authors attempted to compare multiple melody extraction algorithms created since 2005. One of the methods, used also by MIREX, is based on the per-frame comparison, considering different measures:

Voicing Recall Rate - the proportion of frames labelled as melody frames in the ground truth that are estimated as melody frames by the algorithm.

Voicing False Alarm Rate - the proportion of the frames labelled as non-melody in the ground truth that are mistakenly estimated as melody frames by the algorithm.

Raw Pitch Accuracy - the proportion of melody frames in the ground truth for which f_τ is considered correct (i.e. within half a semitone of the ground truth).

Raw Chroma Accuracy - as raw pitch accuracy, except that both the estimated and ground truth f_0 sequences are mapped onto a single octave. This gives a measure of pitch accuracy which ignores octave errors.

Overall Accuracy - this measure combines the performance of the pitch estimation and voicing detection tasks to give an overall performance score for the system. It is defined as the proportion of all frames correctly estimated by the algorithm, where for non-melody frames this means the algorithm labelled them as non-melody, as for melody frames the algorithm both labelled them as melody frames and provided a correct f_0 estimate for the melody (again, within half a semitone of the ground truth).

In Figure 4.11. the authors presented results obtained by the algorithms evaluated at MIREX. To get a general idea of the performance of the algorithms, it is sufficient to focus on two evaluation measures. The raw pitch accuracy, presented in Figure 4.11 a) represents how well the algorithm tracks the pitch of the melody. The overall accuracy on the other hand, as shown in Figure 4.11 b), combines this measure with the efficiency of the algorithm's voicing detection, meaning the voicing-related measures are also reflected in this measure.

As we can see, some collections are generally hard to analyse (for example MIREX09 -5db), in general the collections yield different results for different algorithms. This allows us to spot pros and cons of each approach investigated.

We can also notice that the raw pitch accuracy gradually improved from 2005 to 2009, after which it stayed relatively unchanged. Overall we can see that the average pitch accuracy over a collection lies between 70-80%.

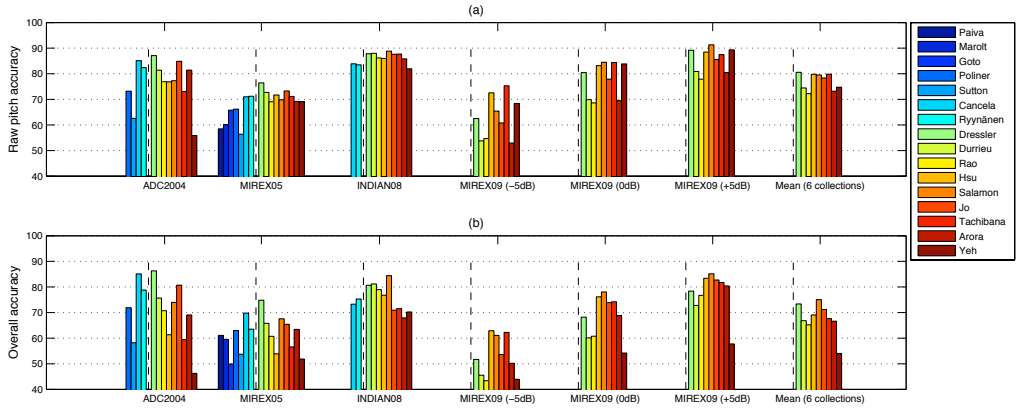


FIGURE 4.11: a) Raw pitch accuracy and b) overall accuracy obtained in MIREX by 16 melody extraction algorithms evaluated in [13]. The vertical dashed line separates the algorithms that were only evaluated on some collections (left of the line) from those evaluated on all six collections (right of the line) [13].

On the other hand, when it comes to overall accuracy, the performance goes down compared to the raw pitch accuracy for all algorithms due to voicing detection being factored into the results. The importance of this step depends on the intended use of the algorithm. Generally, the overall accuracy results lie between 65-70%.

Finally, an important factor in assessment of an algorithm is its complexity. While deriving O-notation is too complex for some of the algorithms, generally it is observed that algorithms involving source separation are significantly more computationally complex than salience based approaches Unfortunately, there is no specific data provided by Salamon and Gómez [9] or by Durrieu [5] on their algorithms.

In conclusion, we believe the solution proposed by Salamon and Gómez is better fitted to the purpose of this project. The paper presents it in a much clearer way and, what is most important, it outperforms the one created by Durrieu significantly, as seen in Figure 4.11. In addition to this, according to tendency it is less computationally expensive, which is quite important when it comes to game designing as we do not want to keep the user waiting for a long time for his level to generate and load.

4.5 Introduction to Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that can be thought of humans' attempt to simulate the brain electronically. Its first conceptual model was developed by Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, in 1943. In their paper, "A logical calculus of the ideas imminent in nervous activity", they describe the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs, and generates an output. Their work served as foundation for designing a computational model based on the brain to solve certain kinds of problems.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Their most common application in computing today is to perform one of these "easy-for-a-human, difficult-for-a-machine" tasks, often referred to as pattern recognition. Thanks to being implemented on computers, they have higher computational capabilities than any human being - calculating a cube of 9124 in memory is not straightforward for us, but a computer can come up with an answer almost immediately. On the other hand, thanks to their structure, they can tackle problems not easy to solve by a simple computer, like facial recognition or regression analysis. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

4.5.1 Models

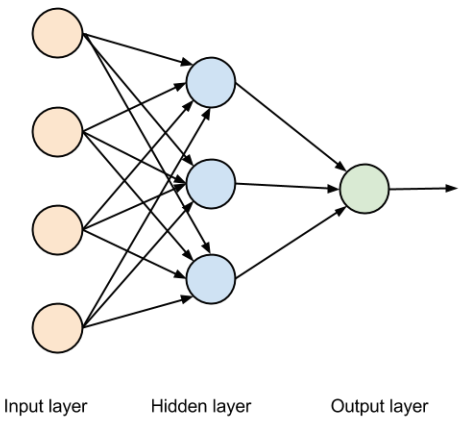


FIGURE 4.12: Diagram of a simple neural network with 4 input nodes, 3 nodes in a hidden layer and one output node.

The computational systems we write are procedural; a program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. On the other hand, neural network are "connectionist" computational systems. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel throughout a network of neurons.

Neural networks are made up of many artificial neurons. There are many different ways of connecting neurons to create a neural network. The number of them depends on a task the network is designed for.

An example system depicted in Figure 4.12 has three layers. The first layer has input neurons (marked red) which send data via synapses to the second layer of neurons (colour blue). Each input into the neuron has its own weight associated with it. A weight is simply a floating point number and they allow us to adjust our network to improve the training outcome. The weights in most neural nets can be both negative and positive, therefore providing excitatory (carrying information) or inhibitory (regulating the activation of excitatory neurons) influences to each input.

As each input enters the nucleus, it is multiplied by its weight. The nucleus then sums all these new input values which gives us the activation. If the activation is greater

than a threshold value, the neuron outputs a signal. If the activation is less than the threshold, the neuron outputs zero. This is typically called a step function.

One type of neural network is called a feedforward network named after the way the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network.

Each input is sent to every neuron in the hidden layer (marked blue) and then each hidden layer's neuron's output is connected to every neuron in the next layer. There can be any number of hidden layers within a feedforward network but one is usually enough to suffice for most problem. There can be any number of neurons in each layer, depending on the problem that is being solved.

4.5.2 Training

Once a network has been structured for a particular application, it is ready to be trained. In the beginning, the initial weights are chosen randomly. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

Majority of networks utilise supervised training. Unsupervised training is used to perform some initial characterisation on inputs [21].

Supervised Training

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set". During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Unfortunately, some networks never learn. This could be caused by the input data not containing the specific information from which the desired output is derived. Networks can also fail to converge if there is not enough data to enable complete learning.

Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorising data. To make sure the network is learning significant patterns rather than plainly memorising the data, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training.

If a network simply cannot solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs.

Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation, which we will describe later.

Yet, training is not just a technique. It involves a "feel", and conscious analysis, to insure that the network is not over-trained. Initially, an artificial neural network configures itself with the general statistical trends of the data. Later, it continues to "learn" about other aspects of the data which may be spurious from a general viewpoint.

When finally the system has been correctly trained, and no further learning is needed, the weights can, if desired, be saved. This way it can be used for predicting the output values for new, unseen data.

Unsupervised Training

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organisation or adaption.

At the present time, unsupervised learning is not well understood. This adaption to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field.

4.5.3 Backpropagation Algorithm

A Backpropagation network is a network that learns by example. Given the desired input and output data telling the network what we want it to do, it changes its weights so that, when training is finished, it produces the required output for a particular input. Backpropagation networks are ideal for Pattern Recognition and Mapping Tasks.

The algorithm starts by first setting up all the network's weights to be small random number. Next, the input pattern is applied and the output calculated (called the forward pass). The calculation is likely to give an output which is completely different to what

we passed in as the expected value, since all the weights are random. The error of each neuron is calculated (expected output - actual output). This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the actual input of each neuron will get closer to its expected output. This part is called the reverse pass. The process is repeated again and again until the error is minimal.

Backpropagation algorithm has some problems associated with it. Perhaps the best known is associated with local minima. This occurs because the algorithm always changes the weights in such a way as to cause the error to fall. But the error might briefly have to rise as part of a more general fall. If this is the case, the algorithm will "gets stuck" (because it cannot go uphill) and the error will not decrease further.

There are are several solutions to this problem. One is very simple and that is to reset the weights to different random numbers and try training again (this can also solve several other problems). Another solution is to add "momentum" to the weight change. This means that the weight change this iteration depends not just on the current error, but also on previous changes. For example:

$$W+ = W + Currentchange + (Changeonpreviousiteration * constant) \tag{4.4}$$

4.6 Mood Detection

It is well known that music can convey emotion and modulate mood. That is why the relation between musical sounds and their influence on the listener's emotion has been well studied.

4.6.1 Emotion Classification

Currently, there is no standard method to measure and analyse emotion in music. However, a psychological model of emotion has found increasing use in computational studies.

In 1989, in his publication [22], J. A. Russell noticed that set of emotional dimensions such as displeasure, distress, depression, excitement etc. are interrelated in a highly systematic fashion. He claimed these relationships can be represented in a spacial model in which concepts fall in circle in the following order: pleasure, excitement, arousal, distress, displeasure, depression, sleepiness and relaxation. Depiction of the model is presented in Figure 4.13a.

A somewhat similar model was described in 1989 by R. E. Thayer. Thayer's two-dimensional emotion model offers a simple but quite effective model for placing emotion in a two-dimensional space [23]. In the model, the amount of arousal and valence is measured along the vertical and horizontal axis, respectively.

(A) Model proposed by J. A. Russell. (B) Model proposed by R. E. Thayer [4].

FIGURE 4.13: Diagram of both models for representing emotions.

Diagram 4.13b depicts the relation between valence and arousal values and the moods perceived by people. As we can see, the high arousal is connected to how energetic the music is, whereas valence refers to how positive (or negative) the emotions in the track are.

4.6.2 Related Literature

Some studies have explored the relationship between physiological activity experienced by a listener and perceived emotion, be it facial expression or speech recognition of even heartbeat and respiratory changes [24]. Others have explored the relationship between perceived emotion and the musical/acoustic features themselves.

One of the first publications on emotion detection in music is credited to Feng, Zhuang, and Pan [25]. They employ Computational Media Aesthetics, that is, analysing two music dimensions to detect mood for music information retrieval tasks. The two dimensions – tempo and articulation, are extracted from the audio signal and are mapped to one of four emotional categories; happiness, sadness, anger, and fear. After that, feature called relative tempo is calculated, and after the mean and standard deviation of the feature called average silence ratio in the presented computational articulation model are calculated, a simple Back Propagation neural network classifier is trained to detect mood.

Different approach was applied by Kim and André [24]. To collect physiological data set, a musical induction which leads subjects to real emotional states was used over many weeks. They used four-channel biosensors to measure electromyogram, electrocardiogram, skin conductivity, and respiration changes. From that, they retrieve a wide range of physiological features and are analysed to find the best emotion-relevant ones and correlate them with emotional states. Finally, the classification of four musical emotions

(positive/high arousal, negative/high arousal, negative/low arousal, and positive/low arousal) is performed by using an extended linear discriminant analysis (pLDA).

In publication by Yang, Lin, Su and Chen [4], the authors presented a tool which recognises a mood in a musical track, allowing a user to then choose the song they want to play by deciding on emotions it is supposed to represent. Specifically, the authors formulate music emotion recognition as a regression problem to predict the arousal and valence values (AV values) of each music sample directly. For this purpose, they adopt Support Vector Regression (SVR) as classifier using a number of chosen features.

Potentially, the second approach described seems more appropriate for our project as it allows for better granularity in the melody emotion detection and, hence, wider variety of changes in the game's environment. However, this area of the project is left to be further researched.

4.7 Song Structure Retrieval

Automatic music structure analysis from audio signals is an interesting topic that receives a lot of attention these days. The technique can be used for music data analysis, indexing, retrieval and management. It decomposes a song into several sections and detects repetitive patterns since they are often semantically meaningful parts.

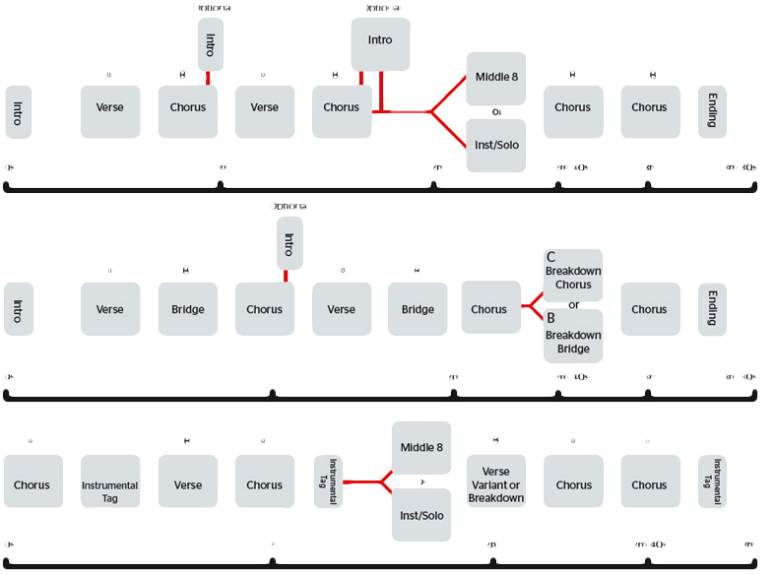


FIGURE 4.14: An example diagram of a song structure.

4.7.1 Song Structure

Popular music is typically created using sectional, repeating forms. Most pop/rock songs have a standard structure: an introduction followed by alternating verses, choruses, and

solos/bridges segments, and concluding with an outro. A musical boundary can be defined as the point in time where a song transitions between two of these segments.

However, this is not always the case as there are different types of musical structure. To describe them, let us represent the verse as A and the chorus as B.

Songs could be strophic - where all the verses are written to the same music, which could be represented as AAA... . Many folk and popular songs represent a strophic structure [26], including "Barbara Allen", "Erie Canal", and "Michael Row the Boat Ashore".

Another song structure seen in the pop culture is thirty-two-bar form, where the structure of each repeated part is made up of four eight bar sections, in an AABA pattern. An example of a song with such a structure could be The Rolling Stones' "Brown Sugar" or The Police's "Every Breath You Take".

In contrast to thirty-two-bar form, which is focused on the verse (contrasted and prepared by the B section), in verse–chorus form it is the chorus that is highlighted (prepared and contrasted with the verse). Good example could be a song "Be My Baby" first recorded by the Ronettes, where the structure could be represented by ABABB(B).

4.7.2 Similarity Matrix

A similarity matrix is a matrix of scores that represent the similarity between a number of data points. Each element of the similarity matrix contains a measure of similarity between two of the data points. Similarity matrices are strongly related to their counterparts, distance matrices and substitution matrices. Similarity matrices have a wide range of uses, including finding clusters of data points.

Similarity between two points i, j is computed as a distance between the features of the beat indices i and j.There are many ways of computing a distance. Some of the most popular are:

Euclidean - for an n-dimensional space, the distance between two vectors of attributes p and q is:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} \tag{4.5}$$

Cosine = Given two vectors of attributes, p and q, the cosine similarity, $\cos(\theta)$, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} ([_i)^2} \times \sqrt{\sum_{i=1}^{n} (q_i)^2}} \tag{4.6}$$

Manhattan - distance, d, between two vectors p, q in an n-dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the

projections of the line segment between the points onto the coordinate axes. More formally,

$$d(p,q) = \|p-q\|_1 = \sum_{i=1}^{n} |p_i - q_i|, \tag{4.7}$$

Correlation - distance between p and q, is defined as:

$$d(p,q) = \frac{(p-\mu_p)\cdot(q-\mu_q)}{||p-\mu_p||_2||q-\mu_q||_2} \tag{4.8}$$

where $||\cdot||_2$ stands for the Euclidean distance, μ_x denotes the mean of the feature vector x, and $\cdot$ represents the dot product.

4.7.3 Related Literature

Many other researchers have considered the importance of patterns and repetition in music. This led to development of many algorithms for the task.

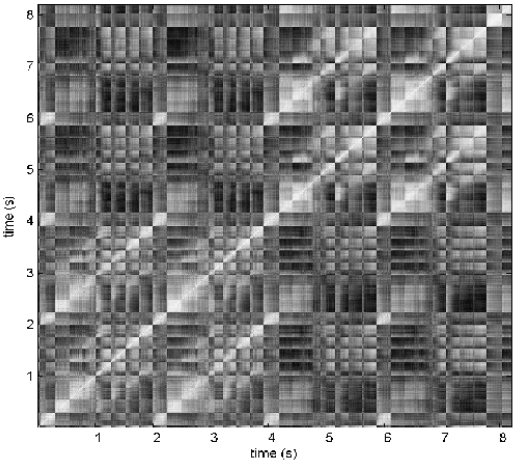


FIGURE 4.15: Self similarity matrix for Bach's Prelude No. 1 in C Major, BVW.

For instance, Foote [27] proposed the use of a self-similarity matrix ("similarity matrix" for short) to visualise similarities between segments of music signals. Given a song, each element of a similarity matrix represents the pairwise similarity between two respective temporal windows of acoustic features. Furthermore, Foote and Cooper [28] developed a music segmentation technique using an audio novelty measure. That is, the local similarity of adjacent musical signals within a coherent section is used to determine section boundaries. Then, all detected sections are compared with each other for their pairwise similarity and clustered into different patterns.

Many researchers developed their methods on top of this solution, approaching the problem of the segments grouping from different angles. Some were using Gaussian Mixture Models (GMM) - a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalising k-means clustering to incorporate information about the covariance structure of the data as well as the centres of the latent Gaussians.

Another approach seen in literature is a variant of Nearest Neighbour Search (NSS), also known as similarity search or closest point search. It is an optimisation problem

for finding closest (or most similar) points. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values. Formally, the nearest-neighbour (NN) search problem is defined as follows: given a set S of points in a space M and a query point $q \in M$, find the closest point in S to q.

Finally, a Non-negative Matrix Factorisation (NMF) can be applied to group the segments, as presented by Kaiser and Sikora [29]. Non-negative matrix approximation is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorised into (usually) two matrices W and H, with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices easier to inspect. Also, in applications such as processing of audio spectrograms non-negativity is inherent to the data being considered. Since the problem is not exactly solvable in general, it is commonly approximated numerically.

Nieto and Jehan [30] based their approach on the NMF method, extending it by adding a convex constrain that results in weighted cluster centroids, representing the different sections of a musical piece in a more effective manner. In standard NMF, matrix factor $W \in \Re_{+}^{m \times k}$, i.e., W can be anything in that space. Convex NMF [11] restricts W to a be convex combination of the input data vectors $(v_1, \cdots, v_n)$. This greatly improves the quality of data representation of W. Furthermore, the resulting matrix factor H becomes more sparse and orthogonal. Then they efficiently extract music boundaries by clustering the decomposition matrices, which take into account the repeated parts across the song in- stead of just detecting sudden local changes.

Some researchers to join the grouping and boundary identification into one algorithm. For instance, Levy and Sandler used Hidden Markov Models (HMM), a generative probabilistic model, in which a sequence of observable X variable is generated by a sequence of internal hidden state Z. The hidden states can not be observed directly and the transitions between hidden states are assumed to have the form of a (first-order) Markov chain. They can be specified by the start probability vector Π and a transition probability matrix A. The emission probability of an observable can be any distribution with parameters Θ_i conditioned on the current hidden state (e.g. multinomial, Gaussian). The HMM is completely determined by Π, A and Θ_i.

In their paper [31], Peeters, La Burthe, and Rodet presented an algorithm which makes use of k-means clustering. The k-means clustering is a method of vector quantisation, originally from signal processing. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells (a partitioning of a plane into regions based on distance to points in a specific subset of the plane). The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum.

An alternative to using any variants of SSM is using supervised learning. For instance, Turnbull and Lanckriet [32] developed a set of difference features that indicate when there are changes in perceptual aspects (e.g., timbre, harmony, melody, rhythm) of the music. They used multiple individual difference features to detect boundaries of a song.

4.8 Level Generation

It is not really surprising that there is no current literature on the problem of automatically generating Guitar Hero buttons given an arbitrary piece of music. However, we believe an algorithm can be developed where the buttons can be mapped to the f_0 in the main melody extracted by main melody extraction algorithm.

Chapter 5

Design and Implementation

In this chapter we will go over the implementation process of the project. We will describe various choices we made, justifying them in the context of our objectives.

First we will describe our solution to the mood detection problem. We first try to determine which musical features are the most correlated to the AV values of the music's emotion. Then, by training a neural network with data containing chosen features we will create a way of determining the arousal and valence value of any musical track, which will be later used in the implementation of our game. In addition to this, by investigating the impact of different parameters we will make sure out network has as good performance and accuracy as possible.

Next, we will move on to main melody detection by looking at two algorithms - one using source separation based approach and the other using the salience based approach. We will evaluate performance of both of them on data from recent pop culture to determine their performance and fitness in this project.

The next section will describe our attempt to automated music segmentation.

Last, but not least, we will talk about the game itself, its architecture, flow of use and design choices made.

5.1 Mood Detection

A common reason for engaging in music listening is that music is an effective means of conveying and evoking emotions. Although they may be subjective, based in part on the listener's cultural and musical background or preferences, there are commonalities in perceived emotion across different listeners based on the characteristics of the music. Several studies have attempted to predict emotion conveyed during music listening. In our approach, we decided to represent the emotion connected to the music using a two-dimensional space with valence on the x-axis and arousal on the y-axis, first proposed by R. E. Thayer [23].

As we described in Section 4.6.1, there is a relation between valence and arousal values for a musical track and the moods perceived by people. In essence, the high arousal is connected to how energetic the music is, whereas valence refers to how positive (or negative) the emotions in the track are.

5.1.1 Choice of Features

Using Essentia library [33], we implemented an extractor to retrieve certain features from a song, which we would expect to have certain impact on the perceived mood of a musical piece:

average loudness - dynamic range descriptor. It rescales average loudness into the [0,1] interval on a per window basis. The value of 0 corresponds to signals with large dynamic range, 1 corresponds to signal with little dynamic range. This could indicate the level of the arousal, with higher loudness implying higher arousal value. We believe this relation could be quite intuitive - sad or peaceful songs tend to be quiet whereas excited or angry emotions are usually linked to louder tracks.

means and derivatives of variance of rates of silent frames in a signal for thresholds of 20, 30 and 60db. We believe that the values could influence the arousal levels, as the more and the bigger the silent gaps, the sadder / more peaceful the track seems to be, implying the low arousal value. When examining multiple musical tracks we have noticed that the happier or angrier songs can also have such silent gaps, but they tend to be much shorter.

dynamic complexity - computed on 2 second windows with 1 second overlap. The dynamic complexity is the average absolute deviation from the global loudness level estimate on the dB scale. It is related to the dynamic range and to the amount of fluctuation in loudness present in a recording. We believe this feature would have an impact on both examined values. However, similarly to the loudness level, arousal should be influenced more - as more dynamic songs (excited or angry) are more likely to suffer from loudness changes, whereas more phlegmatic ones (sad or peaceful) tend to keep the same dynamic complexity level.

BPM - beats per minute value according to detected beats. This feature should be correlated with the arousal level - intuitively, the faster the song, the more energetic it seems.

spectral centroid - centroid statistics describing the spectral shape. It indicates where the "center of mass" of the spectrum is. Perceptually, it has a robust connection with the impression of "brightness" of a sound - an indication of the amount of high-frequency content in a sound. Timbre researchers consider brightness to be one of the perceptually strongest distinctions between sounds [34], and formalise it acoustically as an indication of the amount of high-frequency content in a sound. That is why we believe the spectral centroid might be related to both valence and arousal.

spectral RMS (root mean square) - in physics it is a value characteristic of a continuously varying quantity, such as a cyclically alternating electric current or a sound. It is obtained by taking the mean of the squares of the instantaneous values during its duration or a cycle. This is linked to the loudness of the sound. This is why we believe that it might have an impact on arousal, but we do not exclude its impact on valence.

spectral energy - the energy Es of a continuous-time signal x(t) defined as:

$$Es = \langle x(t), x(t) \rangle = \int_{-\infty}^{\infty} |x(t)|^2 dt \tag{5.1}$$

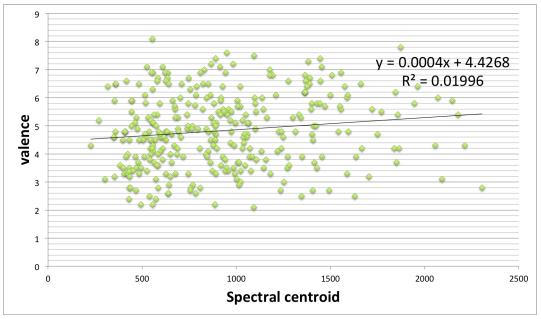
Signal energy is always equal to the summation across all frequency components of the signal's spectral energy density. There have been some research focusing on relation between spectral energy and singing voice. In particular, in their paper [35], S. Ferguson, D. T. Kenny and D. Cabrera were investigating the relation between the value and the experience of male singers. This makes for an interesting case worth considering in our research.

mean and derivative of variance of beat loudness - spectral energy computed on beats segments of audio across the whole spectrum, and ratios of energy in 6 frequency bands. We suspect that the low value of the beat loudness could imply a low arousal.

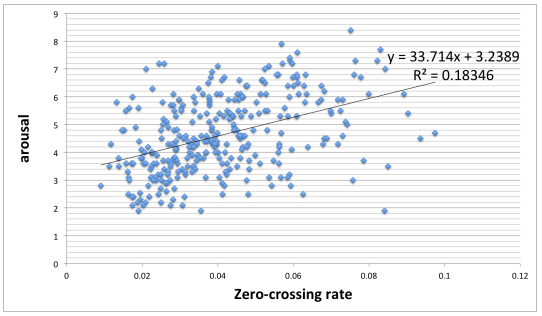
key and its scale estimated key and its scale (major or minor) using Temperley's profile. In music theory, the term key is used in many different and sometimes contradictory ways. A common use is to speak of music as being 'in' a specific key, such as "in the key of C major or in the key of F#". Sometimes the terms 'major' or 'minor' are appended, as 'in the key of A minor' or 'in the key of B major'. Broadly speaking the phrase 'in key of C' means that C is music's harmonic centre or tonic (the first degree of the scale, or the root of the scale). The terms 'major' and 'minor' further imply the use of a major scale or a minor scale. Thus the phrase 'in the key of E major' implies a piece of tonal music harmonically centred on the note E and making use of a major scale whose first note, or tonic, is E. We believe that those features can have an impact on both arousal and valence - songs performed in minor scale are traditionally connected to being sad, whereas the major scale is usually linked to positive emotions.

scale and key of the chords taken as the most frequent chord, and scale of the progression, whether major or minor. Scale commonly known to have a big influence on our perception on music [36]. It seems to be mostly the result of cultural conditioning as when people listen to tunes, they rely heavily on their memory. Such constant stimulus to our musical memory helps to generate expectations of what might come next in a tune or preserve the sound - emotion relation.

means of zero-crossing rate - the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in music information retrieval, being a key feature to classify



(A) A graph representing a correlation between spectral centroid and valence values.



(B) A graph representing a correlation between zero-crossing rate and arousal values.

FIGURE 5.1: Chosen results of bivariate correlation with multiple regression.

percussive sounds. We believe it could be related to the arousal value. Music has a fairly normal distribution of frames with lower and higher zero-crossing rates. Speech however displays a much more skewed distribution. This could have an impact on songs where the vocals are quite rapid and energetic, for example rap music, and therefore might have a significant impact on mood recognition in our system. ZCR is defined formally as:

$$ZCR = \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbb{I}\{s_t s_{t-1} < 0\} \tag{5.2}$$

pitch salience of a spectrum - given by the ratio of the highest auto correlation value of the spectrum to the non-shifted auto correlation value. Unpitched sounds (non-musical sound effects) and pure tones have an average pitch salience value close to 0 whereas sounds containing several harmonics in the spectrum tend to have a higher value. We think the value could have an effect on both the valence and arousal as pitch salience is often described as the probability of noticing a tone or clarity or strength of tone sensation.

mean and derivative of variance of sensory dissonance (to distinguish from musical or theoretical dissonance) of an audio signal given its spectral peaks. Sensory dissonance measures perceptual roughness of the sound and is based on the roughness of its spectral peaks. Given the spectral peaks, the algorithm estimates total dissonance by summing up the normalised dissonance values for each pair of peaks. These values are computed using dissonance curves, which define dissonance between two spectral peaks according to their frequency and amplitude relations. Dissonance could be related to low valence.

5.1.2 Correlation Between Features and Mood Perception

In our exploration we decided to base our research on data collected in "1000 Songs for Emotional Analysis of Music" music library [37], to avoid personal bias in assessing

the mood of the song. The songs in the dataset were annotated by more than 300 crowdworkers on Amazon Mechanical Turk. Each song was annotated for arousal and for valence separately.

As a first step towards understanding the pattern by which audio features might account for emotion ratings, we conducted correlational analyses between features and mean valence/arousal ratings from the data set. We performed a bivariate correlation analysis with the valence/arousal ratings as the dependent variable, and each of the 22 features as the explanatory variable. Example of the results we achieved can be seen in Figure 5.1, the rest are included in Appendix A (Chapter 8, Section 8.1) for reference.

We found significant correlation between **valence** and derivative of variance and mean *silence60*, derivative of variance of *silence30, dynamic complexity, spectral centroid, spectral RMS, spectral energy, zero-crossing rate, pitch salience, and both mean and derivative of variance (dvar) of dissonance.*

For **arousal**, we noticed correlation with *spectral centroid, pitch salience, zero-crossing rate*, both *mean* and *dvar* of *silence60, spectral energy, mean dissonance* and *dynamic complexity.*

Values of all the features were then normalised between 0 and 1 to prepare them for the neural network training.

5.1.3 Neural Network for Mood Prediction

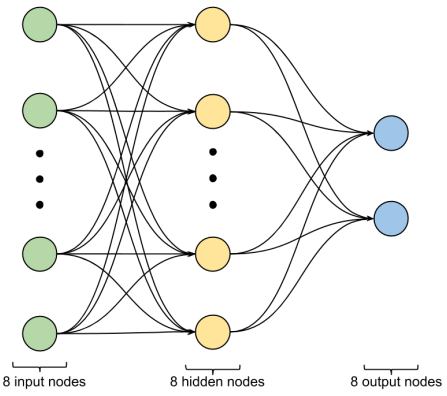


FIGURE 5.2: A diagram depicting the structure of our artificial neural network for mood detection.

Our goal was to train the network to predict mean participant valence and arousal values for musical excerpts. Our first network implementation was a supervised, feedforward network with backpropagation. The input consisted of normalised values of 8 features: *spectral centroid, pitch salience, zero-crossing rate, silence60 mean and dvar, mean dissonance, dynamic complexity* and *spectral energy.* The network had two outputs - arousal and valence.

As all the training data was normalised, the input and output values were within a range of 0 to 1. The training set consisted of 50 input and output arrays. Each input array had 8 values, one per audio feature, and its corresponding output array had the two desired arousal and valence values.

The network's task was to provide the valence and arousal values based on the 13 audio features. The output values fell within a range of 0 to 1. Since desired outputs

were average valence/arousal ratings provided by participants on a scale from 1 to 9, the network outputs were rescaled back. The training set consisted of eight input and output arrays. Each input array had 13 values, one for each audio feature, and its corresponding output array had the two desired arousal and valence values. The connection weights from input to the hidden nodes and from hidden nodes to the output ones were initialised to random numbers.

The network was built, trained, and tested using the pyBrain python library for neural network implementation.

We trained our network for 1000 epochs with many different sizes of the hidden layer and default values for all the other parameters. The performance based on that can be seen in Table 5.1.

Hidden neurons are the neurons that are neither in the input layer nor the output layer. Using additional layers of hidden neurons enables greater processing power and system flexibility at the cost of additional complexity in the training algorithm. Having too many hidden neurons can be thought of as a system of equations with more equations than there are free variables: the system is over specified and incapable of generalisation. Having too few hidden neurons, conversely, can prevent the system from properly fitting the input data, and reduces the robustness of the system.

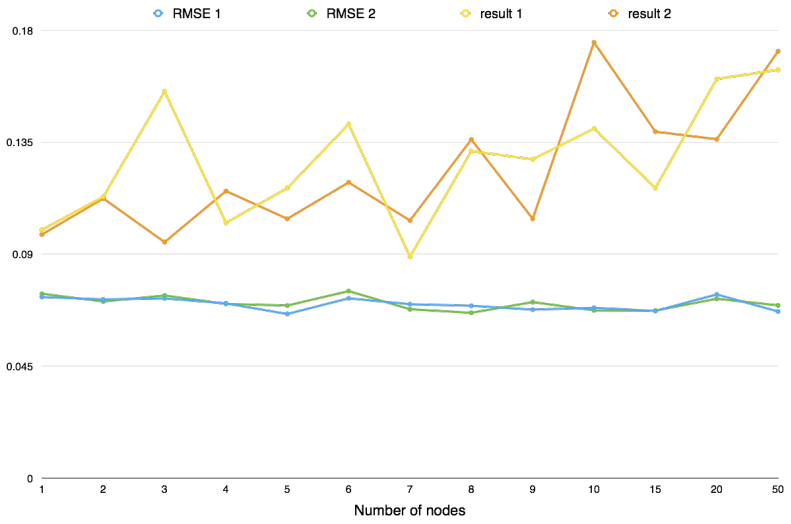


FIGURE 5.3: Data presented in table 5.1, plotted on a diagram.

As we can see, the optimal solution is the one with 7 nodes in the hidden layer. Although the initial RMSE returned after training is not overall minimum, all the values - so both the training ones and the ones after the evaluation, are local minimas and one of the minimal values overall. This decision can be justified by the fact that although for some cases we managed to achieve smaller RSME from the training, the network was in fact overfitting, and doing really well for the already known input, but worse for a new one. To avoid overfitting the network, we kept the number of hidden units equal to the number of input units.

Having found the optimal number of nodes in the hidden layer, we moved on to find the learning rate parameter. Training parameter that controls the size of weight and bias

No. of Nodes	RMSE 1	RMSE 2	result 1	result 2
1	0.0727638005274	0.0740582536152	0.0998088934575	0.0978822145006
2	0.071796654024	0.0709793303052	0.113046836083	0.112405435125
3	0.0722212571658	0.0733605257262	0.155412522783	0.0948392717258
4	0.0702013899702	0.0699921976435	0.102602437509	0.115373051966
5	0.0659433293266	0.0693361162261	0.116558760273	0.10423200269
6	0.0722427034758	0.0751383013205	0.142248432275	0.118843096333
7	0.0698701385354	0.0678483277007	0.088954243616	0.103537259056
8	0.0692459138916	0.066424019477	0.131412928439	0.136098090028
9	0.0676910853628	0.0707274913708	0.128139548772	0.104231713578
10	0.0684398705278	0.0673887116962	0.140505458102	0.175156506583
15	0.0671656450239	0.0673141803371	0.116563143115	0.139265837027
20	0.0737978227013	0.0720620813131	0.160424096589	0.136210925296
50	0.0669456166054	0.0694139442297	0.164132829293	0.171603556123

TABLE 5.1: Table showing the root mean square error for training the network for given number of nodes in the hidden layer.

Learning Rate	RMSE	result RMSE
0.3	0.0707970888752	0.14578838717
0.25	0.0699336891245	0.163193322998
0.2	0.0667986974361	0.15521882498
0.15	0.0724218948598	0.104971086068
0.1	0.0684257582616	0.100719004205
0.05	0.0695957657331	0.0979349713899
0.01	0.0689460348924	0.090954243616
0.005	0.0724023992534	0.130733683966
0.001	0.079664786995	0.112619882406

TABLE 5.2: Table showing the root mean square error for training the network for given learning rate parameter value.

changes in learning of the training algorithm. In a standard backpropagation, too low a learning rate makes the network learn very slowly,whereas a learning rate that is too high makes the weights and objective function diverge, so there is no learning at all.

We started our search by setting it to 0.3 and reducing it over time. The results we found can be found in Table 5.2. As we can see, the optimal solution seems to be learning rate at value 0.001.

In the end, we came up with the network which can be seen on Figure 5.2.

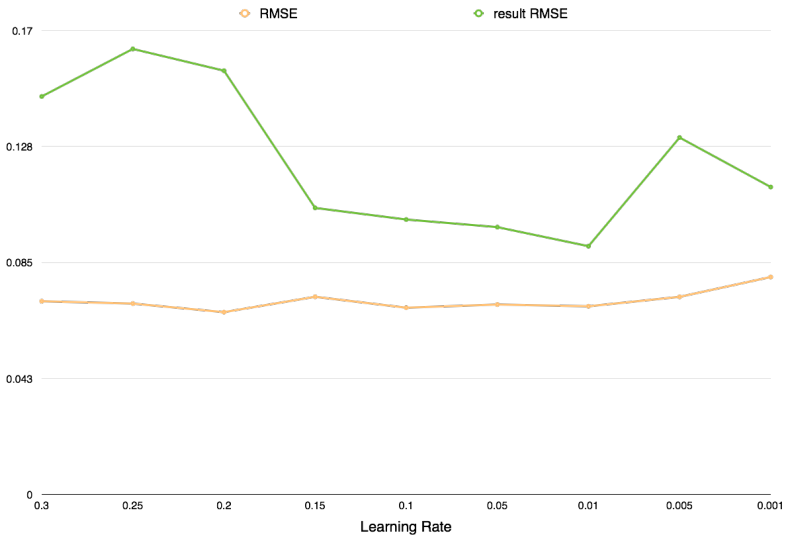


FIGURE 5.4: Data presented in table 5.2, plotted on a diagram.

5.2 Main Melody Extraction

5.3 Structure Retrieval

Understanding the structure of music (e.g. intro, verse, chorus, bridge, and outro) is important as it allows us to divide a song into semantically meaningful segments, within which musical characteristics are relatively consistent.

5.3.1 Feature Choice

To implement a system capable of unsupervised structure recognition, we need to provide it with some data. We investigated two possible values - *Mel-frequency cepstral coefficients* and *harmonic pitch class profile*.

Cepstrum is the result of taking the Inverse Fourier transform (IFT) of the logarithm of the estimated spectrum of a signal. It can be viewed as information about rate of change in the different spectrum bands.

The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.

MFCCs are derived from a type of cepstral representation of the audio clip. The mel-frequency cepstrum differs from cepstrum by having its frequency bands equally spaced

on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

- Take the Fourier transform of a signal.
- Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
- Take the logs of the powers at each of the mel frequencies.
- Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
- The MFCCs are the amplitudes of the resulting spectrum.

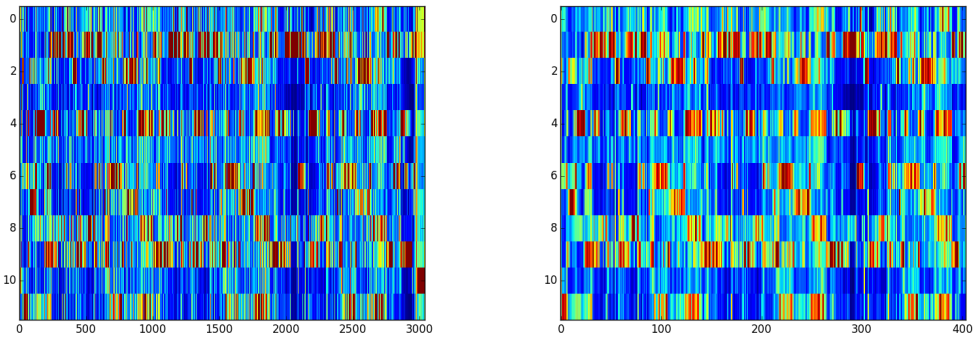
An alternative to using MFCCs as the features to base the algorithm on is *HPCP*.

Harmonic pitch class profiles (HPCP) is a vector of features extracted from an audio signal, based on the Pitch Class Profile descriptor. HPCP is an enhanced pitch distribution feature which is a sequence of chroma - feature vectors describing tonality measuring the relative intensity of each of the 12 pitch classes of the equal-tempered scale within an analysis frame.

HPCP features can be found and used to estimate the key of a piece, to measure similarity between two musical pieces and to classify music in terms of composer, genre or mood. The process is related to time-frequency analysis. In general, chroma features are robust to noise, for example an ambient noise or percussive sounds, independent of timbre and instrumentation and independent of loudness and dynamics.

The General HPCP feature extraction procedure is summarised as follows:

- Input musical signal.
- Do spectral analysis to obtain the frequency components of the music signal.
- Use Fourier transform to convert the signal into a spectrogram. (The Fourier transform is a type of time-frequency analysis.)
- Do frequency filtering. A frequency range of between 100 and 5000 Hz is used.
- Do peak detection. Only the local maximum values of the spectrum are considered.
- Do reference frequency computation procedure. Estimate the deviation with respect to 440 Hz.



(A) Example of a chromagram without any further enhancement.

(B) Example of a chromagram after beat-synchronisation.

FIGURE 5.5: Harmonic pitch class profiles chroma features calculated for a song by The Beatles- "Help!".

- Do Pitch class mapping with respect to the estimated reference frequency. This is a procedure for determining the pitch class value from frequency values. A weighting scheme with cosine function is used. It considers the presence of harmonic frequencies (harmonic summation procedure), taking account a total of 8 harmonics for each frequency. In order to map the value on a one-third of a semitone, the size of the pitch class distribution vectors has to be equal to 36.
- Normalise the feature frame by frame dividing through the maximum value to eliminate dependency on global loudness.

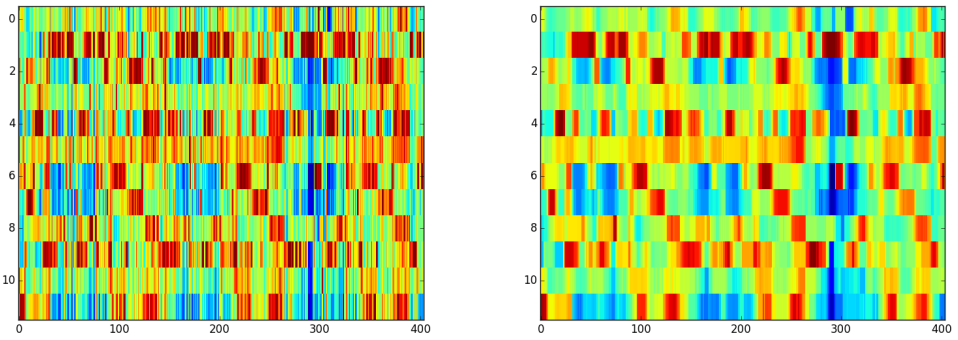
The discussion of results given each of the alternatives are discussed in Section **??**.

5.3.2 Feature Preparation

In this section, we will describe the process of preparation of the features for improving the performance of the algorithm. For simplicity and clarity, when talking about the features, we will first focus on analysis based on HPCPs, followed by one on MFCCs. We decided to investigate both possibilities as they present the track from completely different perspective. For example, the HPCP chroma might fail to distinguish vocal and instrumental parts if the underlying harmonic patterns are exactly the same. On the other hand, when working with MFCCs we expect the opposite behaviour - good performance on parts that are different in terms of timbre.

Harmonic Pitch Class Profiles

A series of transformations are applied to the data in order to distinguish the different parts of a song more efficiently with preserving the accuracy..



(A) Chroma feature after applying log normalisation.

(B) Chroma feature after applying sliding median filter of size h=9.

FIGURE 5.6: Beat-synchronised chroma created for song "Help!" by The Beatles with applied enhancements.

First, we need to we synchronise our data with the beats detected in the musical track. This process allows to reduce local variation by summarising (usually taking the mean or a median) frame-wise features that occur between two beats, yielding fewer but longer beat-synchronous frames. The rationale for doing so is that many features, such as chord labels that occur between two consecutive beats tend to be the same. Thanks to focusing on the values of the features on a per-beat basis, we manage to largely normalise variations in tempo. However, the main advantage of applying the beat-synchronisation is that we manage to reduce the amount of data to analyse, and hence, the size of the matrix, we are operating on.

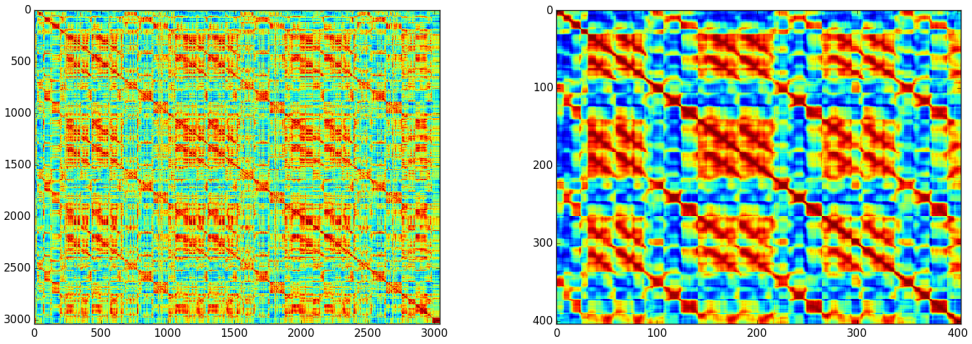
This leads to beat-synchronous chromagrams. A diagram of a chromagram after beat synchronisation can be seen in Figure 5.5b. As we can see, the size has decreased dramatically, which makes the segmentation process computationally cheaper.

Following the beat-synchronisation, we apply log normalisation to the chroma feature. This allows us to reduce the effect the outliers from the trend will have and further improve the contrast between the related and unrelated beat frames. The enhancement achieved by applying log normalisation can be seen in Figure 5.6a.

As the next step, we applied a sliding median filter of size h is run against each of the beat-synchronous and log-normalised chromagram channels, which can be seen in Figure 5.6b. Thanks to the median filter, we can come up with sharper edges than with a regular mean filter. This becomes really useful in obtaining section boundary precision.

By filtering features across time, we retain the most prominent chromas within the h-size window and remove smaller artefacts, which are irrelevant in our con- text. The Figure **??** presents the chromagram after applying the sliding median filter.

We then proceed to compute the Self Similarity Matrix (SSM) of the pre-filtered beat-synchronous chromagram. The SSM is essentially a pairwise comparison of a given set of features using a specific distance measure between the features of the two beat indices



(A) Similarity matrix generated from harmonic pitch class profiles chroma without further enhancement.

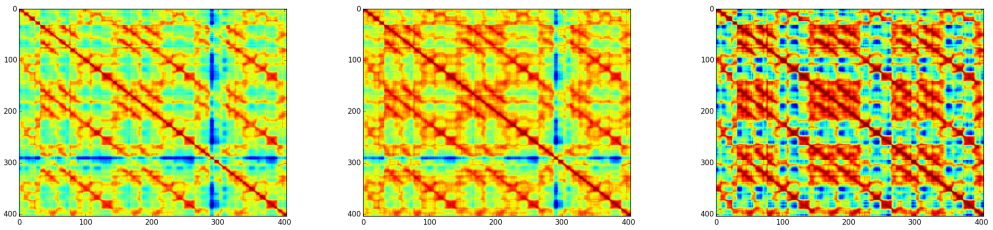
(B) Similarity matrix generated from beat-synchronised harmonic pitch class profiles chroma.

FIGURE 5.7: Comparison of SSM generated from unprocessed and enhanced chromas, using correlation distance.

i and j. The result of every such comparison is stored in a N x N symmetric matrix D, such that D(i, j) contains said distance. In particular, D(i, j) stores the same value ad D(j, i), and for every i D(i, i) is equal to 0.

We investigated the influence of the type of the distance calculated on the SSM produced for the enhanced chroma. In our research we looked into four types of distance: euclidean, manhattan, correlation and cosine. Our results are presented in Figure 5.12. As we can see in Figures 5.12a and 5.12b, the contrast achieved is much weaker. Not only there are fewer blue spots signifying small or even no similarity between points, but the amount of points that are significantly similar is also reduced.

When we look at the SSM computed using cosine distance, we can notice that the amount of the similar points has increased, more similar to the the one generated using the correlation distance. However, the correlation distance on Figure 5.7b contains more dark blue spots, implying that it exposes more beats that are, in fact, not similar. This

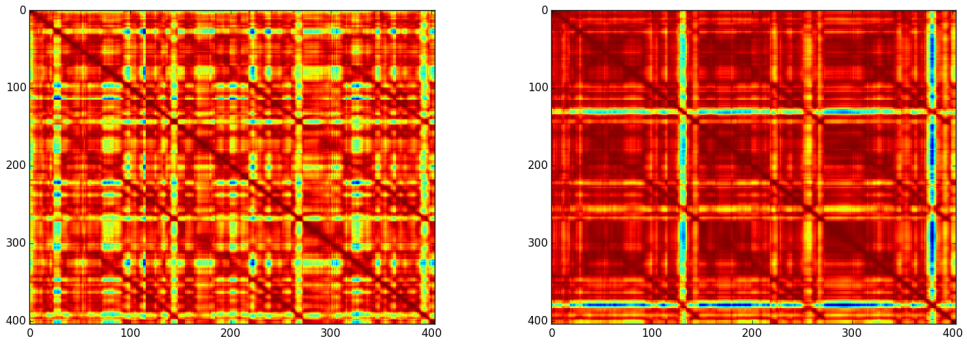


(A) Similarity matrix calculated from harmonic pitch class profiles chroma using Euclidean distance.

(B) Similarity matrix calculated from harmonic pitch class profiles chroma using Manhattan distance.

(C) Similarity matrix calculated from harmonic pitch class profiles chroma using cosine distance.

FIGURE 5.8: Comparison of SSM computed using different distance formulas. The SSM calculated using correlation distance can be seen in Figure 5.7b.



(A) Similarity matrix generated from Mel-frequency Cepstral Coefficients without normalisation.

(B) Similarity matrix generated from Mel-frequency Cepstral Coefficients with application of log normalisation.

FIGURE 5.9: Comparison of SSM generated from unprocessed and enhanced MFCCs, using correlation distance.

is why, in our design of the structure retrieval of a song we decided to use SSM computed using correlation distance.

Mel-frequency Cepstral Coefficients

Similarly to the case of Harmonic Pitch Class Profiles, we start the preparation of the features by beat-synchronisation to decrease the size of the data for further analysis.

Now we have to determine whether log normalisation will improve the clarity of the SSM. As we can see in Figure 5.9, the use of log normalisation could decrease the amount of segments found, as more similar points are exposed.

Finally, we compute the SSM. Again, we investigated the possibility of generating it using Euclidean, Manhattan and cosine distances. The diagrams presenting our findings can be seen in Appendix B (9). Similarly to when we were working with HPCPs, the correlation distance gave us the most contrasted, sharper images.

The result of this process can be seen in Figure 5.9b.

5.3.3 C-NMF

We can view the SSM as an array of column vectors where each vector corresponds to a window. Suppose we have a set of vector templates. Vectors in the steady regions of a song may be directly found in the set, while vectors in the boundary regions may be approximated by linear combination of vector templates. Making this observation, we believe the Non-negative Matrix Factorization (NMF) could be useful in our situation.

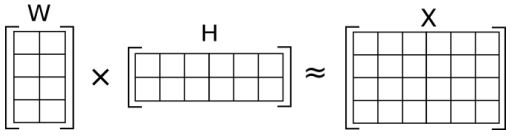


FIGURE 5.10: Illustration of approximate non-negative matrix factorization: the matrix X is represented by the two smaller matrices W and H.

In NMF, the $N \times N$ self similarity matrix X is approximately factorised into product of a $N \times k$ matrix W, can be interpreted as a cluster row matrix, and $k \times N$ matrix H, composed of the indicators of these clusters, where k is the rank of the composition. This can be described as $X \approx WH$. The jth column of W can be viewed as the vector template for the ith segment type. The jth column of H describes the intensities of the kth segment types for the jth window. In NMF, both W and H are enforced to be positive (i.e. X must be positive too). We denote a row vector by $\boldsymbol{z}$ and a column one by $\boldsymbol{z}^T$.

However, in data mining, sometimes it can be beneficial to ensure X to contain meaningful "cluster centroids", i.e., to restrict W to be convex combinations of data points. C-NMF adds a constrain to $W = (\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, \ldots, \boldsymbol{w}_k^T)$, such that its columns $\boldsymbol{w}^T$ are, in fact, convex combinations of the features of X:

$$\boldsymbol{w}_j^T = \boldsymbol{x}_1^T f_{1j} + \boldsymbol{x}_2^T f_{2j} + \ldots + \boldsymbol{x}_N^T f_{Nj} \qquad j \in [1:k] \tag{5.3}$$

The linear combination is convex if all coefficients f_{ij} are positive and the sum of each set of coefficients $\boldsymbol{f}_j^T$ must be 1. Formally, this can be represented as: $f_{ij} \geq 0, f_{ij} = 1$

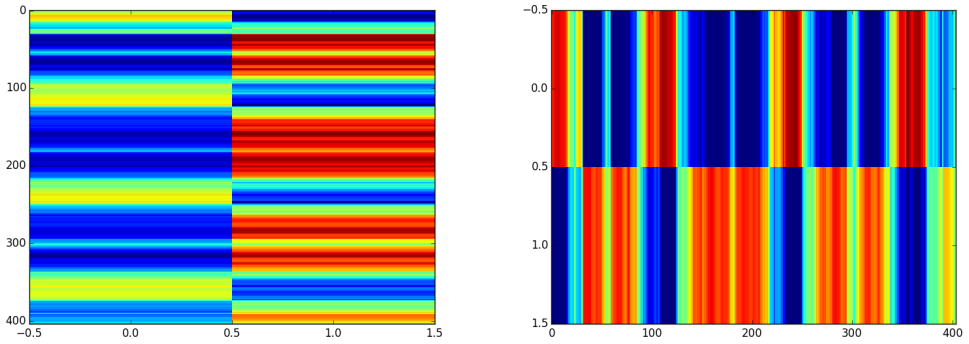
This results in $W = XF$, where $F \in \mathbb{R}^{N \times k}$, which makes the rows $\boldsymbol{f}_i$ interpretable as weighted cluster centroids. The decomposition matrices R_j, are obtained as follows: $R_j = \boldsymbol{w}_j^T \boldsymbol{h}_j$, where $j \in [1:k]$. Finally, C-NMF can be formally characterised as: $X \approx XFH$.

In C-NMF, the matrix W is a set of convex combinations of the rows of the input matrix X, which contrasts with NMF, where no such constraint exists. This means that, each row x_i represents similarity of the time frame i with the rest of the time frames, storing information about the time frae i across the entire song.

By computing the C-NMF we separate basic structural parts. In the next sec- tion, we describe how the factorization via C-NMF relates to structure and show how we can use that result for music structure discovery. ========== change ==========

==== maybe try to get the plots of the decomposition matrices? =====

Apart from this, another important benefit of C-NMF over NMF is that matrices F and G become naturally sparse when adding the convex constrain. In case of NMF the G does not always become sparse. Thanks to that, when using C-NMF we are more likely to find similar decomposition matrices for the same input than NMF, which is more sensitive to its initialisation [30].



(A) The cluster matrix W.

(B) The activation matrix H.

FIGURE 5.11: The result of C-NMF computed for "Help!" by The Beatles with rank k = 3.

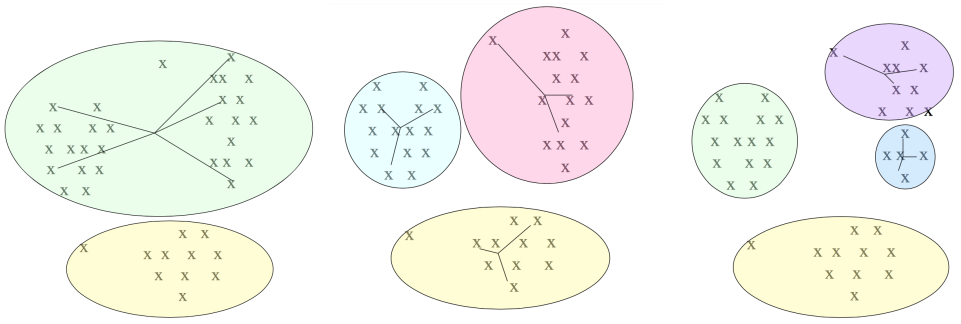
5.3.4 Boundaries

In this section we will investigate different ways of obtaining section boundaries from the decomposition matrices obtained from applying C-NMF to the similarity matrix. An example of resulting cluster and activation matrix computed with C-NMF with rank k = 2 can be seen in figure 5.11.

K-means Clustering

Clustering is an unsupervised classification of patterns, for example observations, data items, or feature vectors, into groups called clusters. The points within each cluster should be similar to each other and dissimilar to points belonging to another cluster. The problem has been addressed in many contexts and by researchers in many disciplines.

K-means clustering is considered one of the simplest unsupervised learning algorithms that can solve the well know clustering problem. It follows a simple and easy way of classifying a given data set through a certain number of clusters (assume k clusters).



(A) The k value is too small -- many long distances to centroids.

(B) Good k value - distances to centroids are quite short.

(C) The k value is too big - little improvement in average distance.

FIGURE 5.12: Diagrams depicting impact of the k value on the clustering result [38].

The main idea is to define k centres, one for each cluster. Much care should be put into their placement, as different location of centres causes different result. This is why the most intuitive solution is to put them as far apart as possible. The next step is to take one point after another from the data set and associate it with the nearest centre,

When all the points have been assigned to some centre, k new centroids are calculated as baycentres of the clustering that resulted from the first phase. Once we have calculated the new centroids, a new binding has to be done between the same data set points and the nearest new center. Very often this will result in points moving between different clusters. This can be considered second phase of the algorithms.

Those two phases are repeated in a loop. As a result, we may notice that the k centers change their location step by step until no more changes are done, and the algorithm converges.

We ran k-means clustering with k = 2 to each one of the C-NMF decomposition matrices, interpreting them as row-vector features. We efficiently obtained the section boundaries The choice of k = 2 allows us to detect boundaries (i.e. there's a boundary or not), regardless of how the various sections cluster. However, after comparing the output of an this algorithm with a manually created segmentation, we noticed that k-means clustering's performance was not suited for our use. The granularity of the segmentation was too high - very often it separated parts of verse, or even fractions of seconds. Even after merging values that were close together and getting rid of the smallest segments, the boundaries detected were too granular.

Another Approach

Having applied the C-NMF, we computed, we have generated two decomposition matrices - W, called the cluster matrix, and H, an activation matrix, so that by multiplying them we can recreate the SSM, ie. $X \approx WH$

To obtain the boundaries, we filter both the cluster matrix W and the activation matrix H. This means that for every row corresponding to a frame in which the beat occurs, we find the indexes of the its maximum values. Thanks to this simple clustering, we obtain a assignment of each of the frames to one of the 'types' of segments. By iterating through a matrix generated and such ways and recording the indexes at which a song changes from one portion to another, we manage to obtain section boundaries in an efficient way. In our implementation, we start the computation with rank $k = 3$ and increase it if not enough bounds were found. This is to avoid overfitting and creating tiny segments, for example, one per verse line.

Once we have boundaries, we combine them within a distance window of size h so that boundaries close to each other get merged in their average location.

This method, although much simpler, produced much neater output, with fewer, but more meaningful segments.

5.3.5 Labelling

In our structure retrieval, we investigated different ways of labelling of the segments.

First approach built directly on our way of segmenting the song. Similarly to when attempting to find boundaries between segments, we decomposed our self similarity matrix X with rank increased. This allowed further differentiation between structures, for instance, if certain part of a song was thought of as a chorus, but it is different enough to get separated in a C-NMF of a higher rank.

Once we have decomposed the matrix X, we filtered the clustering matrix W. This way, we computed initial clustering for each beat frame. To synchronise labels calculated in such way, we assigned a numerical label which occurred within each of the interval between two segment to them.

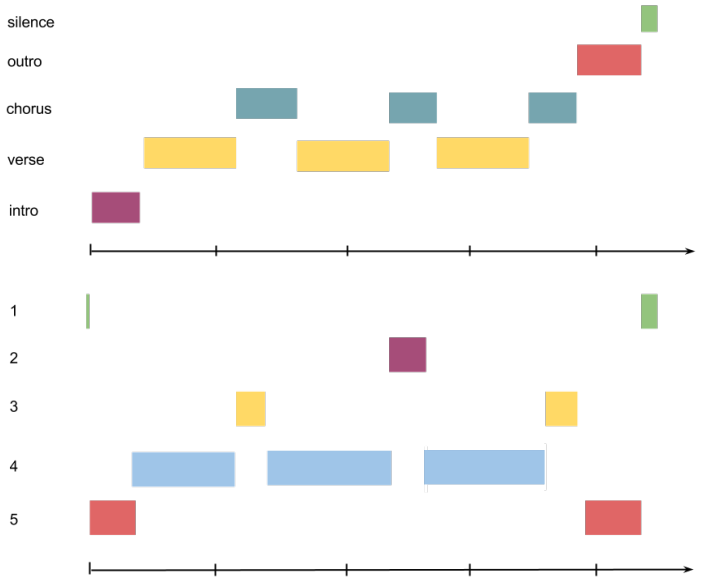


FIGURE 5.13: A depiction of the pure vocals detection as a labelling method (bottom) compared with manual segmentation and labelling (top).

However, having gathered data about the main melody of the song earlier on, we could use it to more accurately predict the labels for each of the song segments, producing labels that easy to understand to a person. In attempt to do so, we synchronise the array of pitches with the beats to get beat synchronised features which we could with ease refer to once we have the boundary frames.

First intuition we had was to investigate the amount of silent frames occurring in each of the segments. We designed heuristics to decide whether the segment we look at is vocal (majority of the frames contain the main melody), semi-vocal (the same amount of frames that are contain the main melody and not), instrumental (minority of the frames contain the main melody) or silent. However, this way alone, we lose the distinction between the vocal parts, for instance, between the verse and the chorus.

A visualisation of our segmentation and labelling using only vocal/instrumental heuristics which can be seen on Figure 5.14. The diagram makes it evident that our boundary

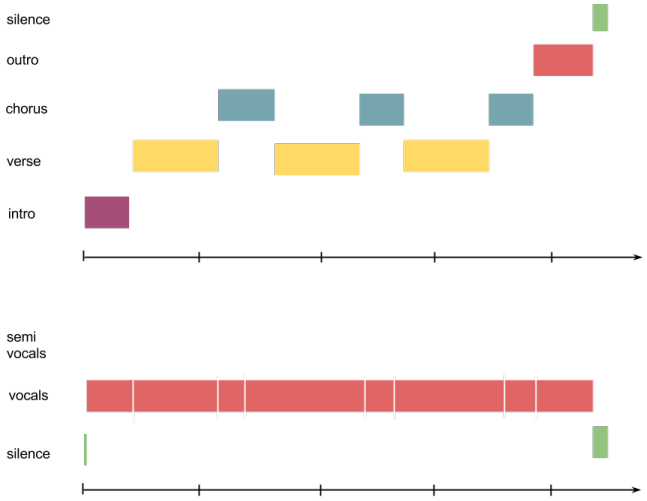


FIGURE 5.14: A depiction of the pure vocals detection as a labelling method (bottom) compared with manual segmentation and labelling (top).

finding works quite well. As we can see, each beginning of the chorus is detected with high accuracy. Moreover, the beginning and the end of the intro and outro parts are predicted really well. In addition to this, the segmentation process manages to detect the silence segments in the beginning and end of track. However, our clustering algorithm detects the end of the chorus too early, absorbing a part of it into the verse that follows.

5.4 The Game

In this section, we will go over the architecture and the design choices made when planning and implementing our game.

The game is written mostly in Swift, a multi-paradigm, compiled programming language created by Apple Inc. for iOS and OS X development. It was first introduced at Apple's 2014 Worldwide Developers Conference (WWDC). Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks, building on the best of C and Objective-C, without the constraints of C compatibility. It adopts safe programming patterns and adds modern features to make programming easier, more flexible, and more fun [39].

We chose this language as we wanted to create a game for the OS X platform. In addition to this, the author also had a personal interest in learning the language.

5.4.1 Data Storage

The game relies on preserving user's scores and the levels generated by them. We need a way of storing them and all the information retrieved when analysing the songs to avoid

regenerating the levels for the same song, for example if the user has a music piece they particularly like.

Core Data is the standard way to persist and manage data in both iPhone and Mac applications. It is an object graph and persistence framework provided by Apple in the Mac OS X and iOS operating systems.

Core Data describes data with a high level data model expressed in terms of entities and their relationships plus fetch requests that retrieve entities meeting specific criteria. Code can retrieve and manipulate this data on a purely object level without having to worry about the details of storage and retrieval.

Core Data allows data organised by the relational entity–attribute model to be serialised into XML, binary, or SQLite stores.

Core Data is also a persistent technology, in that it can persist the state of the model objects to disk. But the important takeaway is that Core Data is much more than just a framework to load and save data - it is also about working with the data while it is in memory. We decided to use Core Data rather than a separate database as our game only needs to store data used by the current user, that will be utilised almost immediately after loading into memory.

The model might cause some intensive memory usage if we decide to create a big amount of users, however, as it is an offline game that can be played on a personal machine, in contrast to web application, the number of users should remain relatively small.

5.4.2 Menu

Although not usually adopted in OS X games, we decided to follow the Model-View-Controller design pattern in implementing our application. We believe it was a right choice as the complexity of the main menu would have to be then supported throughout the played level. This would not only be a performance strain, but would also cause the code to be messy.

When first facing the menu, the user has an option of creating an account, logging in as a user or playing a quick game, not requiring any user data. The quick game is essentially an ability of playing one of the predefined levels, without a choice of creating a new one.

Once the user has created an account or chosen an existing one, they can either follow the level creation or level loading option. If they choose to create a new level, they have to select a file from their hard drive they would like to use as the base for their level. Otherwise, they go to the window, where they can select a level and either play it or remove it from their catalogue.

5.4.3 Level Description

Once we move on to playing a game, the `GameViewController` unpacks the `GameScene` - an object representing a scene of content in Sprite Kit.

Sprite Kit provides a graphics rendering and animation infrastructure that can be used to animate arbitrary textured images, or sprites. It uses a traditional rendering loop where the contents of each frame are processed before the frame is rendered. Its advantage is that it was developed for Apple hardware, hence it is optimised to render frames of animation efficiently using the graphics hardware. Thanks to this, the positions of sprites can be changed arbitrarily in each frame of animation. Sprite Kit also provides other functionality that is useful for games, including basic sound playback support and physics simulation. [40].

In the game scene, there is a set of buttons at the bottom of the screen. Players use the strum bar along with the fret buttons to play notes that scroll down the screen. The Easy difficulty only uses the first three fret buttons, that is, the green, red, and yellow. The Medium difficulty uses the blue button in addition to those three, and Hard and Expert use all five buttons.

The score is calculated based on how many scrolling notes we manage to hit. Every time we hit, the performance bar on the right side of the screen goes up, otherwise it goes down. If it hits the minimum, it the player loses. However, if the player manages to keep the performance level at the maximum for an appropriate amount of time, the number of the points scored for the new notes gets doubled until he misses a note or wins the level.

The player can at any time pause, stop or replay the game. They can also control the volume of the music and other sounds in the game.

Upon completion of the level the player presented with their score, shown as stars and a concrete number. The player can later revisit the levels if they want to improve their score.

5.4.4 Melody Detection as a Game Changer

The main part of the gameplay relies on the user pressing buttons that line up on the screen. As this is a music game, there are various ways of making the process more intuitive and hence attractive to potential players.

One of the possibilities is to use the main melody of the song to determine which buttons to issue for the player by tying in the melody extraction.

...

Having processed the list of pitches in such way, we have prepared the ground for the buttons generation.

The main idea of the game is to mimic the playing an instrument on the computer keyboard. For this purpose we looked to 2 most popular instruments - guitar and piano for inspiration.

When playing the piano, we are presented with a set of keys.

5.4.5 Introduction of The Song Segmentation

5.4.6 Impact of the Mood on the Level

5.5 Main Section 2

Chapter 6

Results and Evaluation

6.1 Quantitative

6.1.1 Evaluation of Mood Detection system

We wanted to use neural networks to calculate valence and arousal ratings of songs using audio features we extract.

We computed the mean error between participant ratings and network-predicted outputs across all segments of all test melodies. The network's performance total RMSE was 0.088954243616 on scale from 0 to 1. or 21.62%. The network predicted at an average accuracy of 78.38% for all 20 segments. The plot of the expected and predicted values can be seen in Figure 6.2. These results are more promising than ones of Yang and Lin citemood, where their R^2 scored 58.3% for arousal and 28.1% for valence.•

Results from the static network indicate that a network can be trained to identify statistical consistencies across audio features abstracted from music and satisfactorily predict valence/arousal values that closely match mean participant ratings.

6.1.2 Melody Extraction Testing

To evaluate our implementation of the melody extraction algorithms we can use the technique used at Music Information Evaluation eXchange, described in section 2.4.3 of the report. In particular, we can compare the performance of our implementation when tested on the samples used during MIREX to the official statistics presented in papers [3, 13].

Another way of evaluating the game is creating a set of songs and generating levels for them. After that a trained Guitar Hero player can play those levels. If the buttons were consistently on time with the notes then the melody extraction and game synchronisation techniques are considered to work.

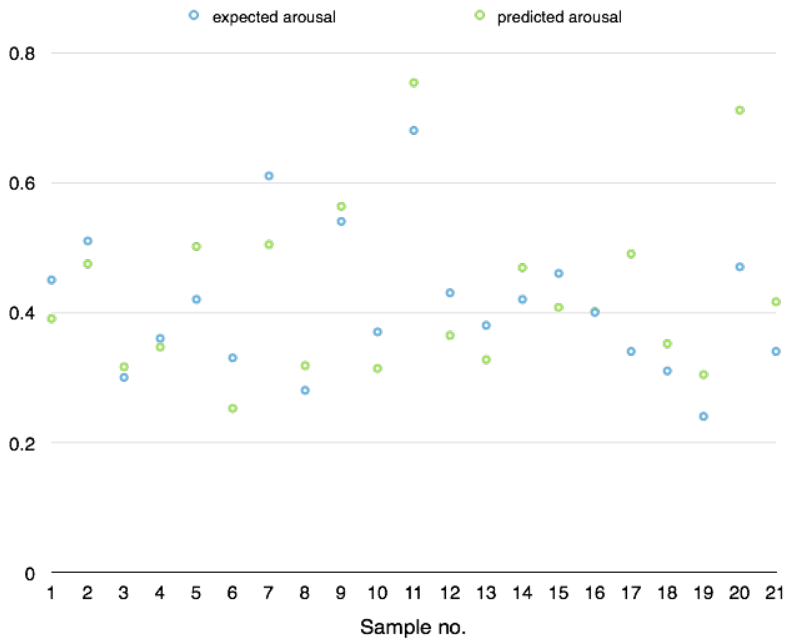


FIGURE 6.1: A plot of the expected and predicted .

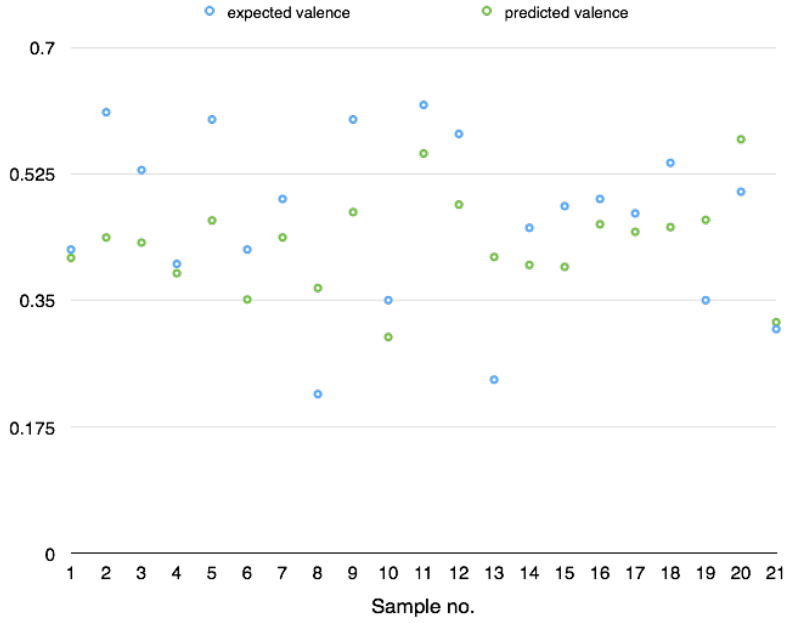


FIGURE 6.2: A plot of the expected and predicted valence.

6.2 Qualitative

Qualitative research is conducted to gain understanding of effects and benefit of the project. They focus on people's own experiences and provide insights and trends in thoughts on a given matter. They can be unstructured or semi-structured, conducted as group discussions, surveys or individual interviews.

expected arousal	expected valence	predicted arousal	predicted valence
0.45	0.42	0.390444	0.408524
0.51	0.61	0.474538	0.436625
0.3	0.53	0.316230	0.429643
0.36	0.4	0.346713	0.387249
0.42	0.6	0.501414	0.460295
0.33	0.42	0.252398	0.350910
0.61	0.49	0.504392	0.436785
0.28	0.22	0.318096	0.366836
0.54	0.6	0.563120	0.471717
0.37	0.35	0.313782	0.298909
0.68	0.62	0.753534	0.552922
0.43	0.58	0.364568	0.482194
0.38	0.24	0.327288	0.409628
0.42	0.45	0.468762	0.398749
0.46	0.48	0.407701	0.396029
0.4	0.49	0.401469	0.454892
0.34	0.47	0.490065	0.444592
0.31	0.54	0.351714	0.451086
0.24	0.35	0.304314	0.461078
0.47	0.5	0.711224	0.572459
0.34	0.31	0.416320	0.319491

TABLE 6.1: Table showing the root mean square error for training the network for given number of nodes in the hidden layer.

6.2.1 Questionnaires

Questionnaires are one of the most common and popular tools to gather data from a large number of people. They generally consist of a limited number of questions that ask participants to rate the effectiveness of various aspects of the activity. The questions should focus on the key points we are trying to evaluate.

Questionnaires tend to be short in order to reduce the amount of time respondents need to complete them, and therefore increase the response rate.

We composed questionnaires that are quantitative and generally consist of close-ended questions (tick the box, or scales), as the open ended questions tent to make data analysis and reporting more difficult.

Before the design and implementation phase of the project, we conducted a study to determine what features could be desirable in the game. We led a survey among ====== FILL IN THE GAP ====== people aged ======== FILL IN THE GAP ====== asking about their past experience with music rhythm games. The questions and the results are presented in the Table 6.2. Each of the questions was answered on a scale from 1 to 5, where 1 is a No, 3 is Neutral and 5 is a Yes.

Question	Average	Std. deviation	Min	Max
Do you like playing games?	2.5	2.5	2.5	2.5
Do you like listening to music?	2.5	2.5	2.5	2.5
Do you often play games?	2.5	2.5	2.5	2.5
Have you ever played Guitar Hero or other rhythm music game?	2.5	2.5	2.5	2.5
Did you feel like the choice of songs was limiting you?	2.5	2.5	2.5	2.5
Were you able to load in your song of choice in there?	2.5	2.5	2.5	2.5
Would you like to be able to load a song into it?	2.5	2.5	2.5	2.5
Did you feel like the game graphics were reflecting the emotions in the song?	2.5	2.5	2.5	2.5
Would you like the game to reflect the emotions in the song?	2.5	2.5	2.5	2.5
Was the game reflecting the section of the song you were in?	2.5	2.5	2.5	2.5
Do you feel it would be useful to know what section of the song is currently played?	2.5	2.5	2.5	2.5

TABLE 6.2: Table presenting the results of the preliminary questionnaire.

We also had to additional questions asking for suggestions, in case there are other features that could be useful to the game or could make the game more attractive that we missed in our initial market research.

What other features did you like in the game? What other features were missing but you feel would be useful in the game?

Chapter 7

Conclusion and Further Work

7.1 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

7.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

7.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

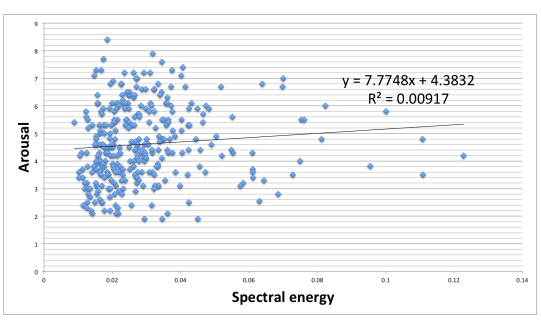
7.2 Main Section 2

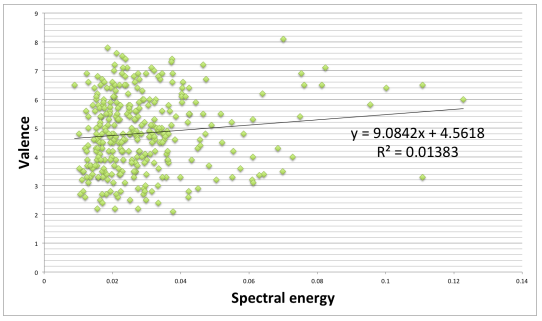
Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

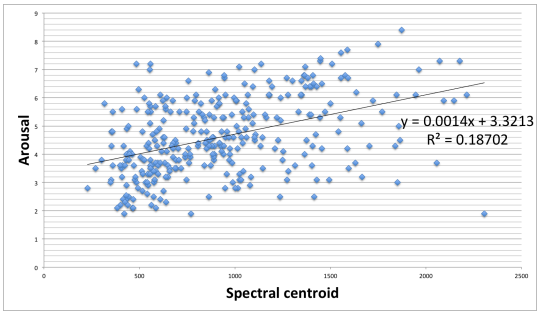
Chapter 8

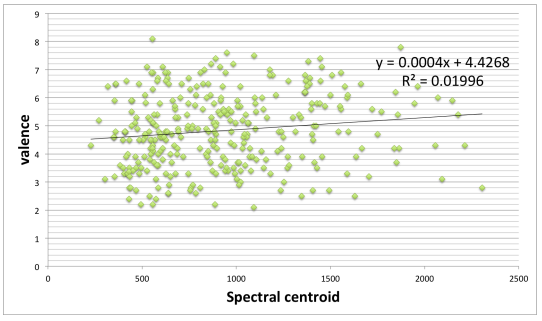
Appendix A: Mood Detection Results

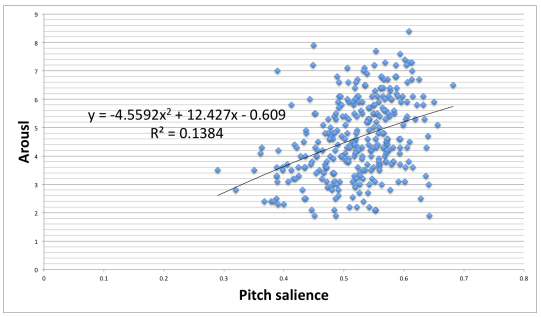
8.1 Bivariate Correlation with Regression

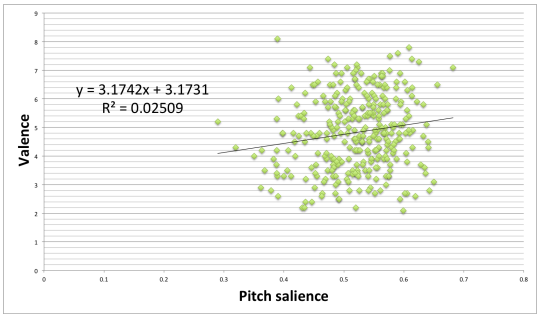


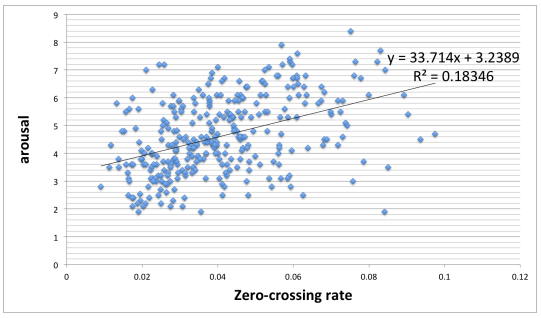




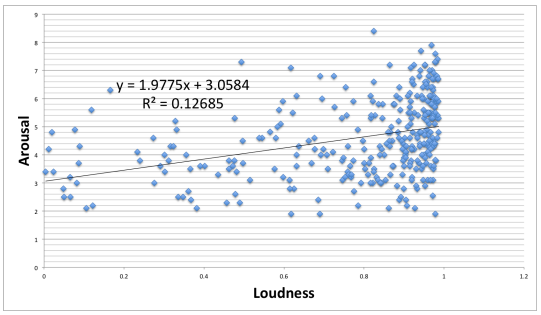


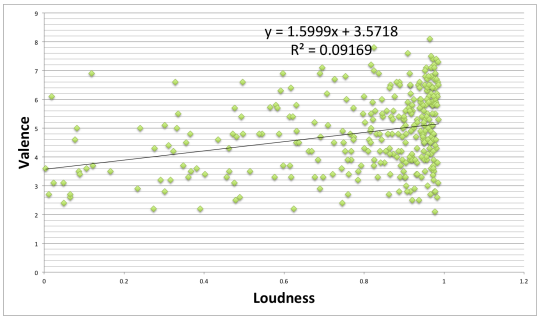

y = -4.5592x² + 12.427x - 0.609
R² = 0.1384
Arousl
Pitch salience

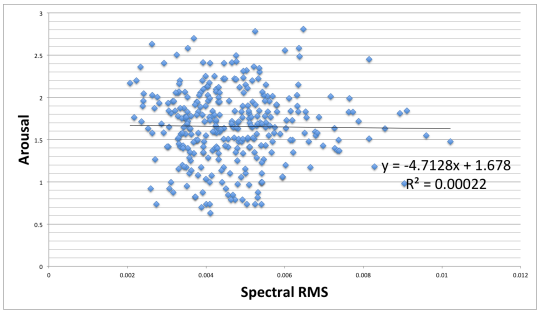

y = 3.1742x + 3.1731
R² = 0.02509
Valence
Pitch salience

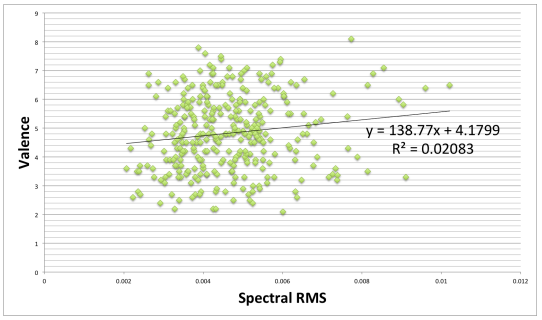

y = 33.714x + 3.2389
R² = 0.18346
arousal
Zero-crossing rate

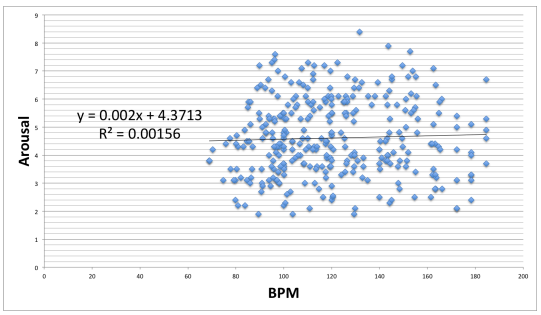

y = 15.783x + 4.1854
R² = 0.0444
Valence
Zero-crossing rate

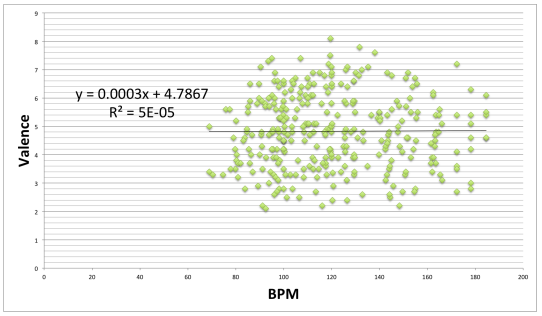

y = 1.9775x + 3.0584
R² = 0.12685
Arousal
Loudness

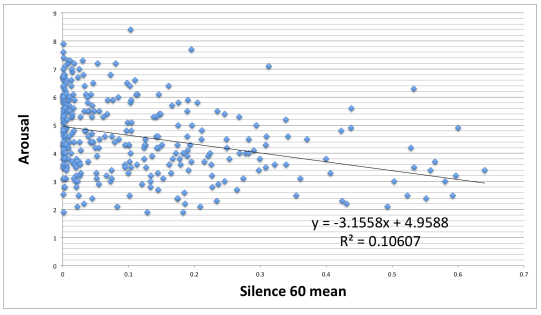

y = 1.5999x + 3.5718
R² = 0.09169
Valence
Loudness

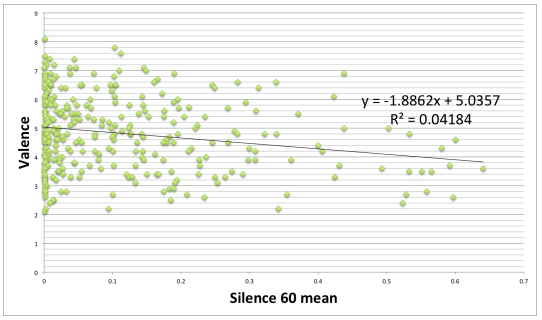

y = -4.7128x + 1.678
R² = 0.00022
Arousal
Spectral RMS

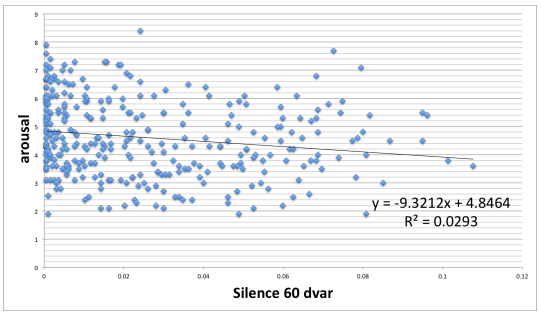

y = 138.77x + 4.1799
R² = 0.02083
Valence
Spectral RMS

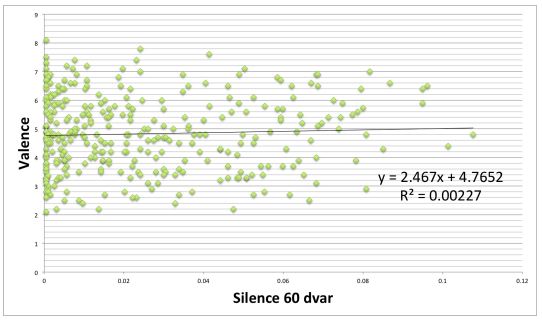

y = 0.002x + 4.3713
R² = 0.00156
Arousal
BPM

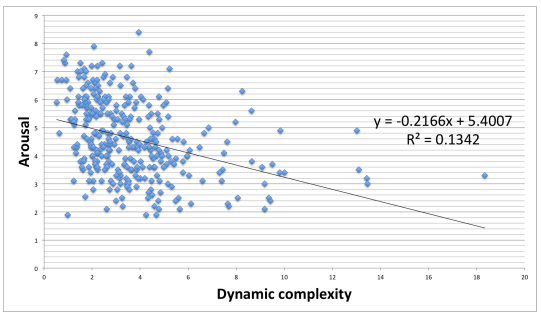

y = 0.0003x + 4.7867
R² = 5E-05
Valence
BPM

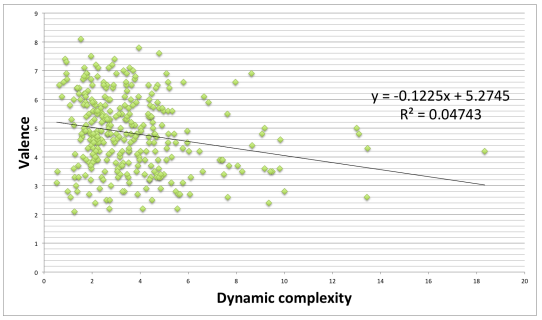

y = -3.1558x + 4.9588
R² = 0.10607
Arousal
Silence 60 mean

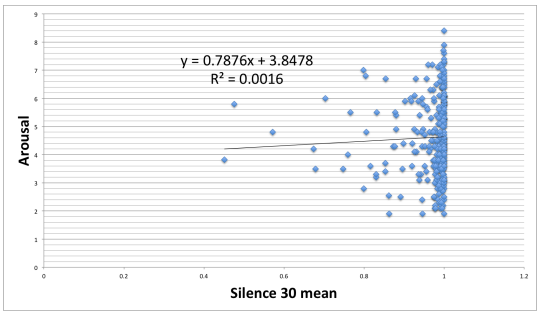

y = -1.8862x + 5.0357
R² = 0.04184
Valence
Silence 60 mean

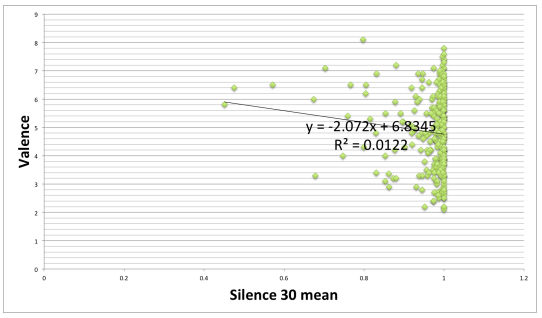

y = -9.3212x + 4.8464
R² = 0.0293
arousal
Silence 60 dvar

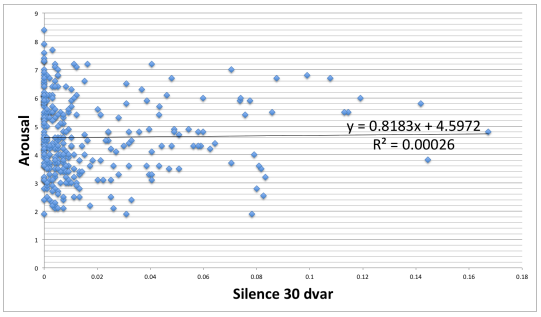

y = 2.467x + 4.7652
R² = 0.00227
Valence
Silence 60 dvar

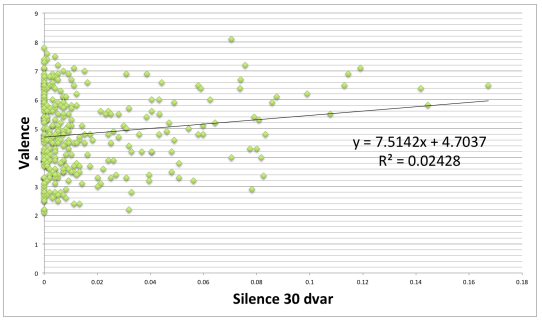

y = -0.2166x + 5.4007
R² = 0.1342
Arousal
Dynamic complexity

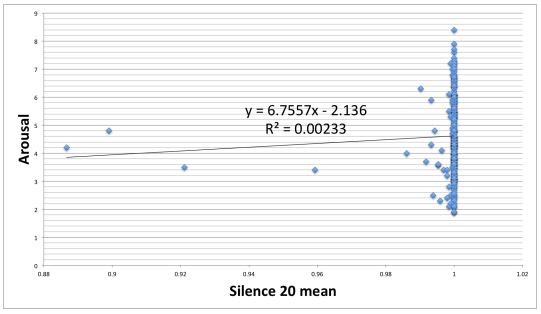

y = -0.1225x + 5.2745
R² = 0.04743
Valence
Dynamic complexity

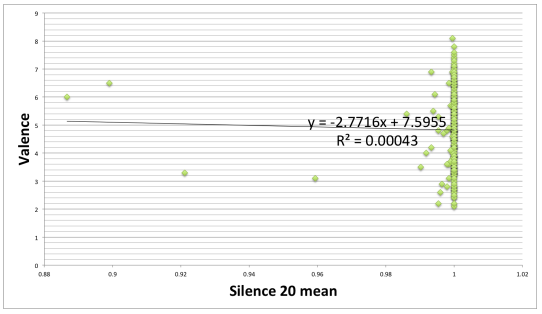

y = 0.7876x + 3.8478
R² = 0.0016
Arousal
Silence 30 mean

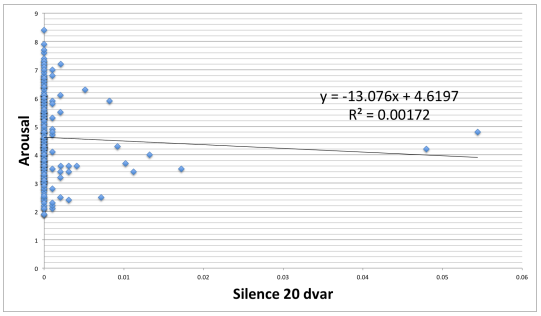

y = -2.072x + 6.8345
R² = 0.0122
Valence
Silence 30 mean

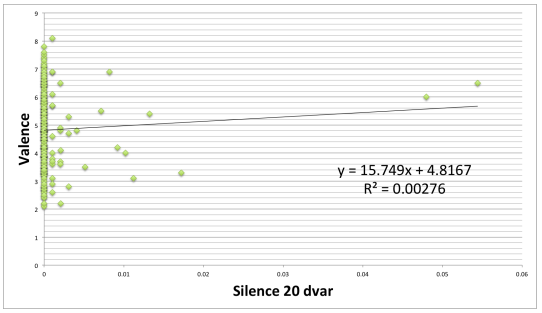

y = 0.8183x + 4.5972
R² = 0.00026
Arousal
Silence 30 dvar

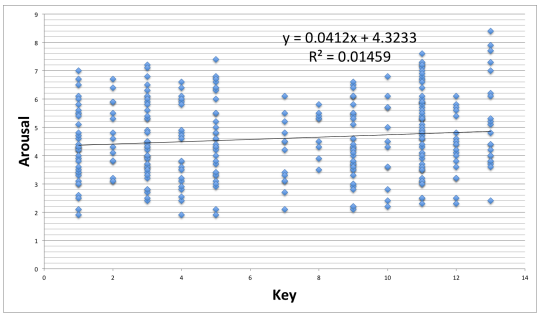

y = 7.5142x + 4.7037
R² = 0.02428
Valence
Silence 30 dvar

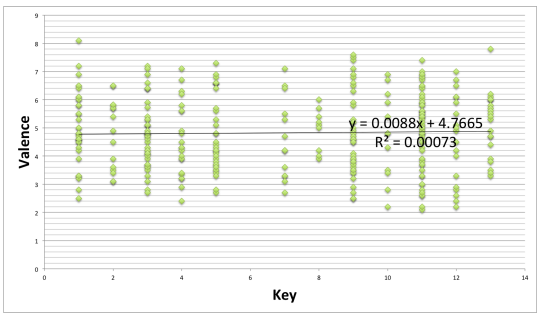

y = 6.7557x - 2.136
R² = 0.00233
Arousal
Silence 20 mean

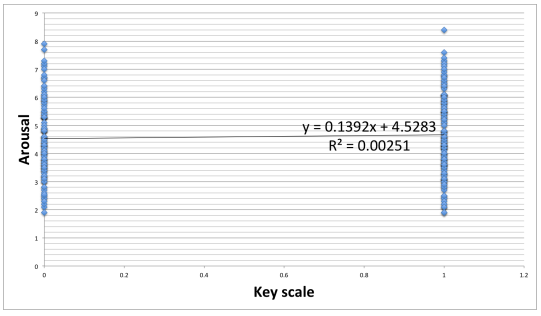

y = -2.7716x + 7.5955
R² = 0.00043
Valence
Silence 20 mean

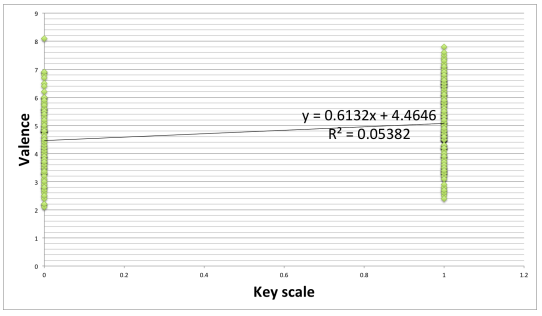

y = -13.076x + 4.6197
R² = 0.00172
Arousal
Silence 20 dvar

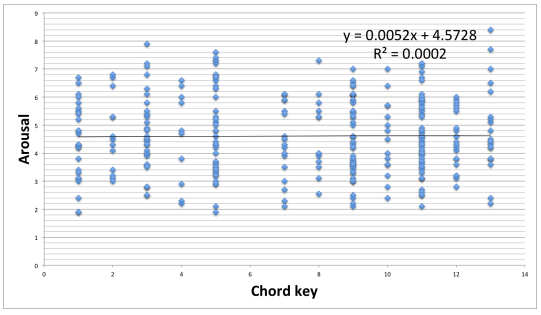

y = 15.749x + 4.8167
R² = 0.00276
Valence
Silence 20 dvar

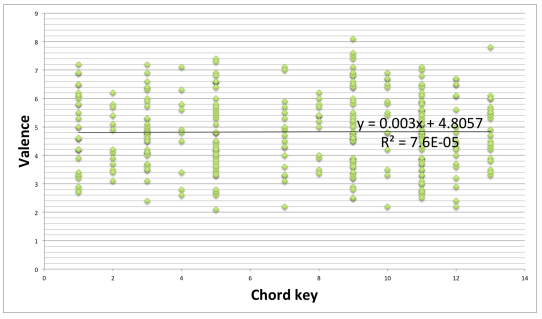

y = 0.0412x + 4.3233
R² = 0.01459
Arousal
Key

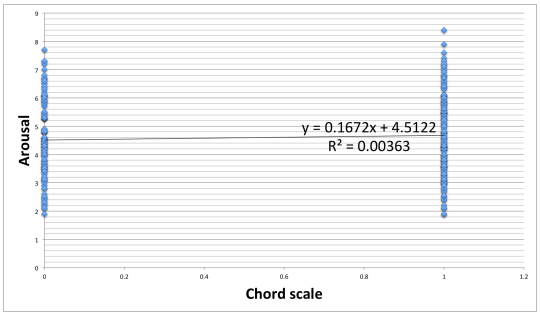

y = 0.0088x + 4.7665
R² = 0.00073
Valence
Key

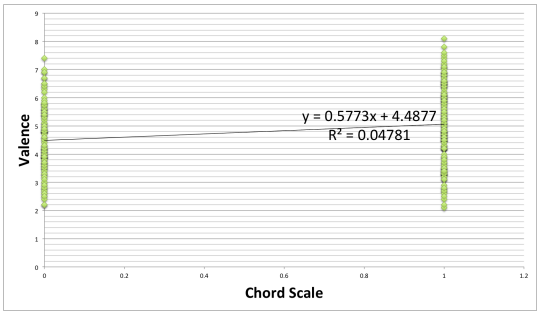

y = 0.1392x + 4.5283
R² = 0.00251
Arousal
Key scale

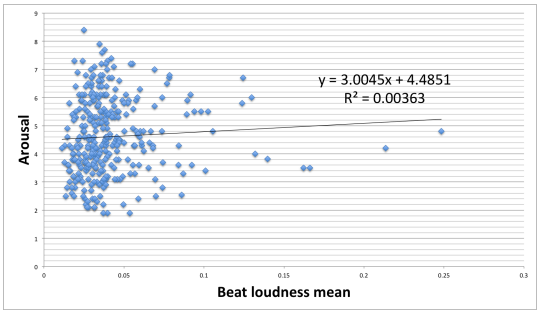

y = 0.6132x + 4.4646
R² = 0.05382
Valence
Key scale

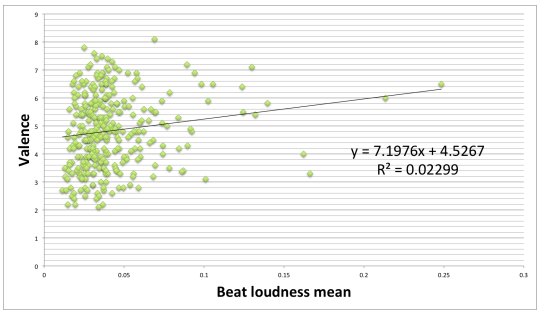

y = 0.0052x + 4.5728
R² = 0.0002
Arousal
Chord key

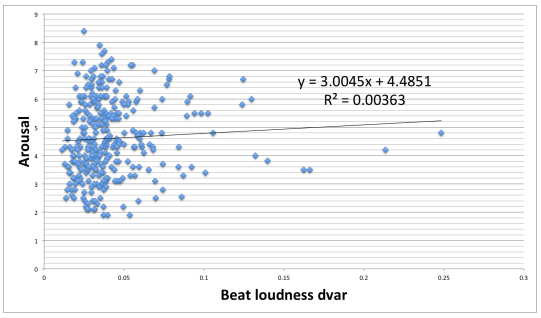

y = 0.003x + 4.8057
R² = 7.6E-05
Valence
Chord key

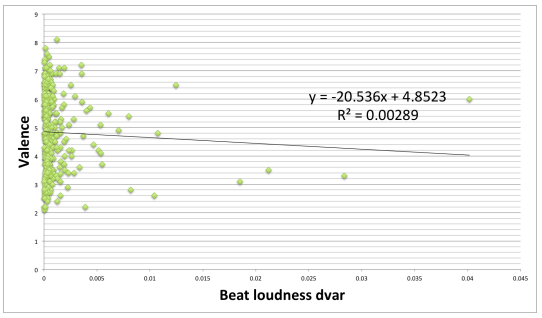

y = 0.1672x + 4.5122
R² = 0.00363
Arousal
Chord scale

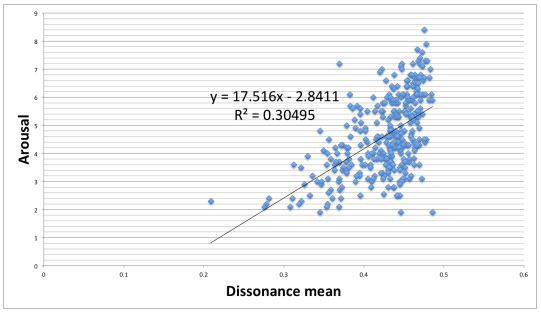

y = 0.5773x + 4.4877
R² = 0.04781
Valence
Chord Scale

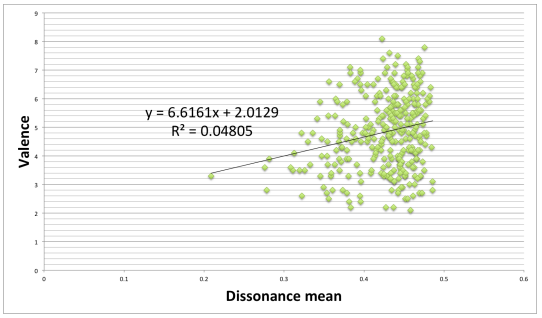

y = 3.0045x + 4.4851
R² = 0.00363
Arousal
Beat loudness mean

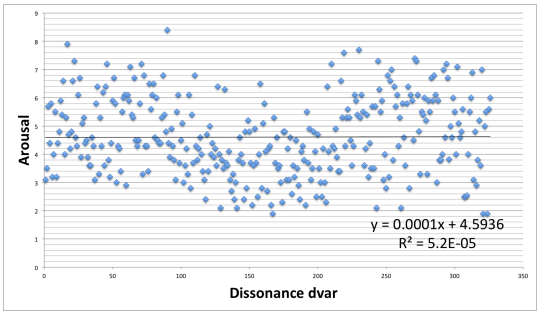

y = 7.1976x + 4.5267
R² = 0.02299
Valence
Beat loudness mean

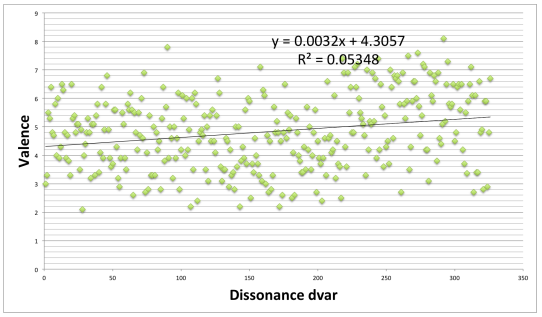

y = 3.0045x + 4.4851
R² = 0.00363
Arousal
Beat loudness dvar


y = -20.536x + 4.8523
R² = 0.00289
Valence
Beat loudness dvar


y = 17.516x - 2.8411
R² = 0.30495
Arousal
Dissonance mean


y = 6.6161x + 2.0129
R² = 0.04805
Valence
Dissonance mean


y = 0.0001x + 4.5936
R² = 5.2E-05
Arousal
Dissonance dvar


y = 0.0032x + 4.3057
R² = 0.05348
Valence
Dissonance dvar

Chapter 9

Appendix B: Structure Retrieval Results

9.1 Structure Retrieval Results

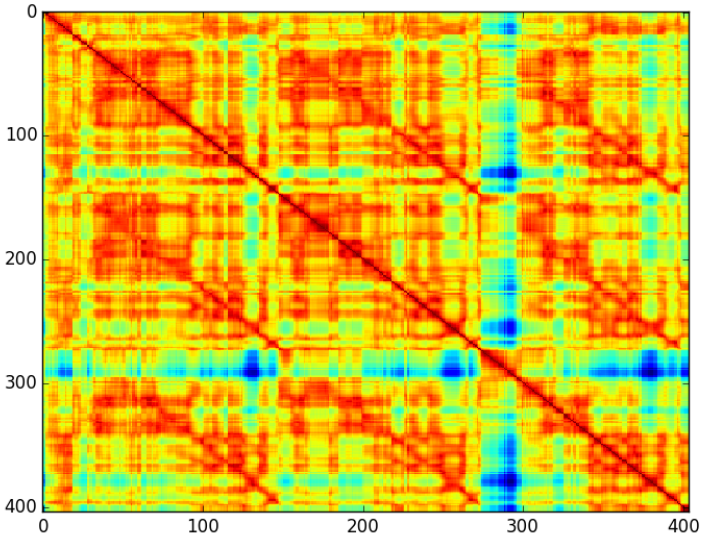


FIGURE 9.1: Similarity matrix calculated from Mel-frequency Cepstral Coefficients using Euclidean distance.

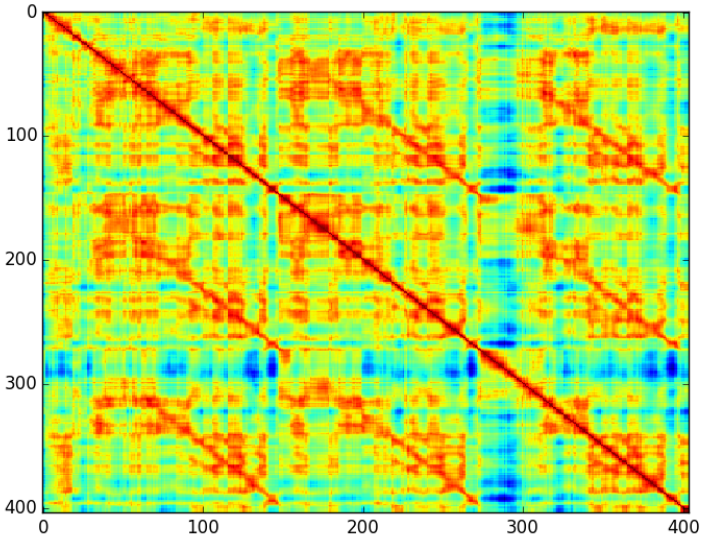


FIGURE 9.2: Similarity matrix calculated from Mel-frequency Cepstral Coefficients using Manhattan distance.

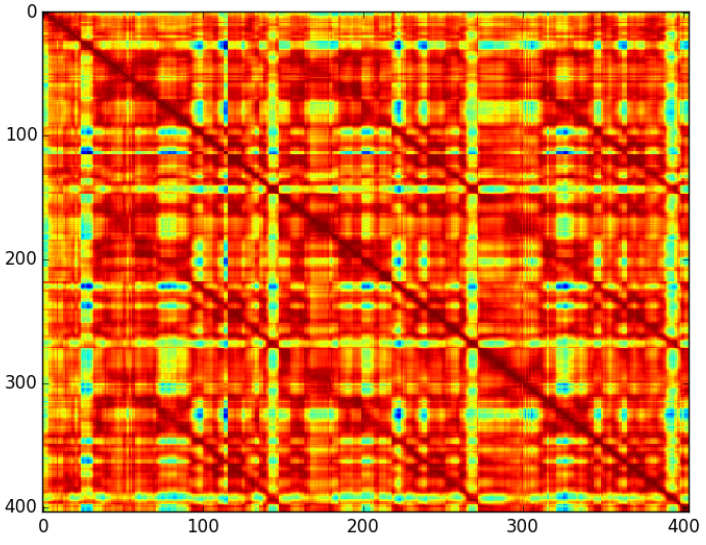


FIGURE 9.3: Similarity matrix calculated from Mel-frequency Cepstral Coefficients using cosine distance.

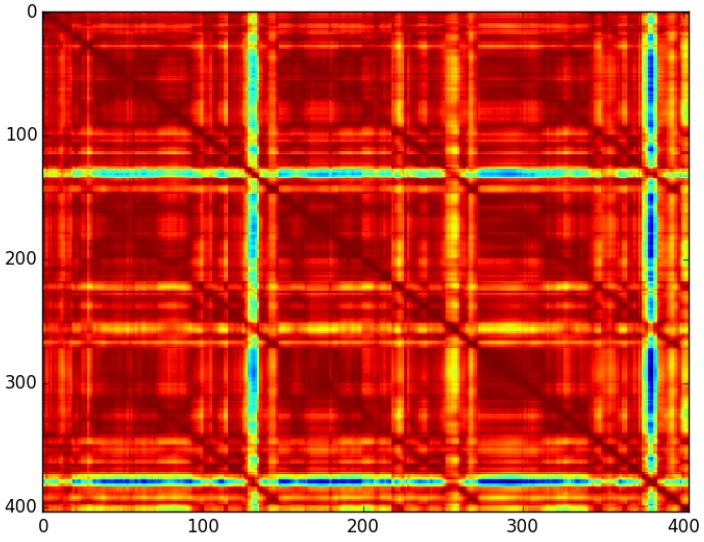


FIGURE 9.4: Similarity matrix calculated from Mel-frequency Cepstral Coefficients using cosine distance.

Bibliography

[1] Ian Bogost. *How to Do Things with Vidoegames*. University of Minnesota Press, 2011.

[2] Guitar hero sales article, 2015. URL http://arstechnica.com/gaming/2009/01/guitar-hero-iii-first-game-to-reach-1-billion-in-sales/.

[3] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Trans. on Audio, Speech and Language Processing*, 20(6), 2012.

[4] Yi-Hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer H. Chen. A regression approach to music emotion recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):448– 457, 2008.

[5] Music video game types, 2015. URL http://en.wikipedia.org/wiki/Music_video_game.

[6] Dance dance revolution screenshort., 2015. URL http://cdn-static.gamekult.com/gamekult-com/images/photos/00/00/34/99/ME0000349967_2.jpg.

[7] Music video game definition, 2015. URL http://psp.about.com/od/pspglossary/a/Music-Game-Definition.html.

[8] Internal section screenshot, 2015. URL http://www.hardcoregaming101.net/internalsection/is2.jpg.

[9] Simtunes screenshot, 2015. URL http://i.ytimg.com/vi/9CLYALO930k/hqdefault.jpg.

[10] Rhythm game definition, 2015. URL http://en.wikipedia.org/wiki/Rhythm_game.

[11] Guitar hero screenshot, 2015. URL https://images-na.ssl-images-amazon.com/images/G/01/videogames/detail-page/gh3.lor.03.lg.jpg.

[12] Guitar hero controller photo, 2015. URL http://www.pixelplayer.se/bilder/texter/rec_8576_1_20081117.jpeg.

[13] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody transcription from music audio: Approaches and evaluation. *IEEE Signal Processing Magazine*, 31(2): 118– 134, 2014.

[14] University of Virginia Mark Whittle. A brief story of how we know about primordial sound, and how we make it audible, 2005. URL http://www.astro.virginia.edu/~dmw8f/BBA_web/unit05/unit5.html.

[15] G. E. Poliner, D. P. W. Ellis, F. Ehmann, E. Gómez, S. Streich, and B. Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Trans. on Audio, Speech and Language Process*, 15(4):564– 575, 2007.

[16] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. URL http://books.w3k.org/.

[17] Short time fourier transform, 2015. URL http://www.originlab.com/index.aspx?go=Products/Origin/DataAnalysis/SignalProcessing/STFT.

[18] J.-L. Durrieu, G. Richard, B. David, and C. Fevotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Trans. on Audio, Speech and Language Process*, 18(3):564– 57, 2010.

[19] Viterbi algorithm, 2015. URL http://en.wikipedia.org/wiki/Viterbi_algorithm.

[20] Vibrato definition, 2015. URL http://en.wikipedia.org/wiki/Vibrato.

[21] aihorizon, 2015. URL http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.html.

[22] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[23] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989.

[24] J. Kim and E. André. Emotion recognition based on physiological changes in music. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (30):2067–2083.

[25] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Music information retrieval by detecting mood via computational media aesthetics. *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 235– 241, 2003.

[26] W. Duckworth. *A Creative Approach to Music Fundamentals*. 2012.

[27] J.Foote. Visualizing music and audio using self similarity. *Proc. of the 7th ACM International Conf. on Multimedia*, pages 77–80, 1999.

[28] J. Foote and M. Cooper. Media segmentation using self-similarity decomposition. *Proc. of SPIE Storage and Retrieval for Multimedia Database*, (5021):167–175, 2003.

[29] F. Kaiser and T. Sikora. Music structure discovery in popular music using non-negative ma- trix factorization. *Proc. of the 11th International Society of Music Information Retrieval*, pages 429–434, 2010.

[30] O. Nieto and T. Jehan. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 236–240, 2013.

[31] G. Peeters, A. La Burthe, and X. Rodet. Toward automatic music audio summary generation from signal analysis. *Proc. of the 3rd International Society of Music Information Retrieval*, pages 94–100, 2002.

[32] Douglas Turnbull and Gert Lanckriet. A supervised approach for detecting boundaries in music using difference features and boosting. pages 42–49, 2007.

[33] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, and et al. Essentia: an audio analysis library for music information retrieval. *International Society for Music Information Retrieval Conference (ISMIR 13)*, pages 493–498, 2013.

[34] D. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, (3):45–52, 1979.

[35] S. Ferguson, D. T. Kenny, and D. Cabrera. Effects of training on time-varying spectral energy and sound pressure level in nine male classical singers. *Journal of Voice*, 24(1):39–46, 2010.

[36] Affective key characteristics. URL http://www.wmich.edu/mus-theo/courses/keys.html.

[37] Soleymani, Mohammad, Caro, Micheal N., Schmidt, Erik M., Sha, Cheng-Ya, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2Nd ACM International Workshop on Crowdsourcing for Multimedia*, CrowdMM '13, pages 1–6, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2396-3. doi: 10.1145/2506364.2506365. URL http://doi.acm.org/10.1145/2506364.2506365.

[38] Clustering algorithms, cs345a: Data mining, stanford university, May 2015. URL http://web.stanford.edu/class/cs345a/slides/12-clustering.pdf.

[39] Swift documentation. URL https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0.

[40] Sprite kit documentation, 2015. URL https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html.