Imperial College London

Department of Computing

Turning Music Into a Game
Tap Along

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 case of music rhythm games, very often the player is limited to the songs preprocessed by the game developers. 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.

We explore the possibility of using main melody extraction to lead the generation of the main task for the user. In addition to this, we investigate a relationship between musical features and a mood perceived in a song to train a neural network capable of predicting arousal and valence values of a melody. Finally, we design and implement a system to detect boundaries between song segments and label them in an easy to understand way.

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, Dr. Iain Phillips, for his sharp insight into the problems encountered, and ability to immediately suggest a good solution to them. His detailed feedback has helped me improve as a scientist and a researcher.

A special thanks to my second supervisor, Dr. Robert Chatley, for providing me with some key pointers on the direction of the project and its potential problems.

Last, but not least, thanks to my parents. For everything.

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, where 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 was soon announced the fastest video game franchise to reach $1 billion in retail sales in the history of the business, with Guitar Hero III being the first single game to reach $1 billion [2].

However, a limited amount of songs transcribed and adjusted to the gameplay 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.

There exist some tools that allow users to manually add new songs to those games, but they are hack-arounds, created by frustrated users, and not real solutions. Moreover, due to the time consuming and difficult nature of the process, most players usually limit themselves to preprocessed 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 create a game which allows them to upload any music track they would like and automatically generate a Guitar Hero-like song corresponding to it.

This is achieved by use 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, a set of contour characteristics, which help distinguish between melodic and non-melodic contours, is defined. This leads to the development of new voicing detection, octave error minimisation and melody selection techniques [3]. Once the pitch of the main melody is estimated, we perform further post-processing to remove the unlikely outliers, as well as adjust the difficulty of the song generated according to our needs.

In this project, we also design and develop an algorithm for mapping the estimated pitches of the main melody 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 attempt to develop a mood extraction system to dynamically generate surroundings in the game. Specifically, we treat music emotion recognition as a regression problem to predict the arousal and valence (AV) values of each music sample directly, which then can be used to generate unique surroundings for every song analysed. 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 problem is tackled by designing and training a neural network to predict listeners' mean valence and arousal ratings associated with musical pieces. Thanks to training on short excerpts, we can then activate the network on parts of the music track uploaded by the user to be able to accurately extract and visualise the mood changes that exist in it.

Moreover, we implement a system to retrieve segments of the music track. This is achieved by identifying the boundaries between parts and labelling them in a way that is clear and understandable to a user. This way we are able to notify the them of their progress in the gameplay, as well as apply more granularity in the mood detection by predicting the emotion on a per segment basis.

This is achieved by generating two self-similarity matrices - one for harmonic pitch class profiles and one for Mel-frequency cepstrum coefficients in a song and decomposing them using Convex Non-Negative Matrix Factorisation and using the results to detect boundaries of a song. The bounds are then used to create a novel algorithm that takes the estimated pitches of the main melody into account for labelling the segments of the song defined by them.

With this project we also 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 songs that also generates the surroundings tailored to every music track.

Main Contributions

1. Evaluation of two main melody estimation algorithms and design of the post-processing system for introduction of variation in difficulty of generated playable songs.

2. Training and use of a neural network for retrieval of arousal and valence values of the music emotion recognition system.

3. Design and implementation of a structure retrieval system which identifies boundaries in the tracks and labels segments defined by them in a way that is clear to a human user.

4. Tap Along - a game allowing users to upload a music track of their choice to generate a custom playable song for them using the methods listed above.

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. In particular, we present an overview of existing solutions to the problems of finding the main melody in a musical track, emotion detection and song segmentation retrieval.

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].

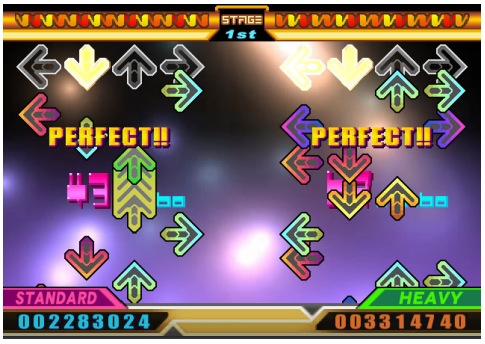


FIGURE 4.1: Screenshot from Dance Dance Revolution, an example of a rhythm music game [6].

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.

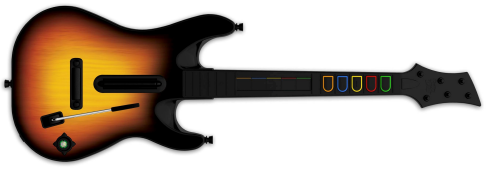
Rhythm games typically focus on dance or the simulated performance of musical instruments, and require players to press buttons in a sequence dictated on the screen. Doing so causes the game's protagonist or avatar to dance or to play their instrument correctly, which increases the player's score [10]. 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 concept of a rhythm-based music game.

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.

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.

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 thought of as an automated extraction of relevant perceptual information (notes, instruments, etc.) from music files (like MP3 or WAV files). First attempted in the 1970s at Stanford University [13], it remains an unsolved problem. The task 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.

In this section we go over the basic terminology used in music analysis.

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).

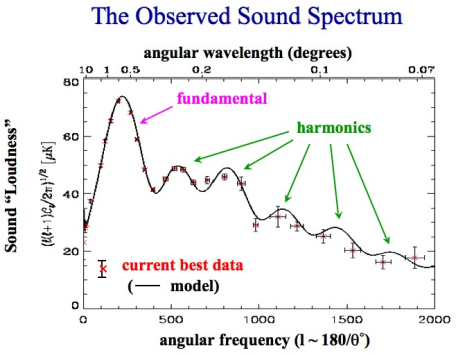


FIGURE 4.4: Example of a sound spectrum diagram [14].

4.3.2 Pitch, Tones, Fundamental Frequency, Timbre

Pitch is the most natural way of ordering sounds on a frequency-related scale. If sounds have a frequency which is clear and stable enough to be distinguished from noise, they can be compared between 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 [15].

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 [3].

Timbre is what makes a particular musical sound different from another, even when they have the same pitch and loudness. For instance, it is the difference between a guitar and a piano playing the same note at the same loudness. Experienced musicians are able to distinguish between different instruments of the same type based on their varied timbres, even if those instruments are playing notes at the same pitch and loudness.

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 [16]. This is usually contrasted with homophony, where musical parts move 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 effect 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 varies 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 [17], "...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 what predominant is. However, it fits well with our project's objective — generating a game level based on changes in the pitch.

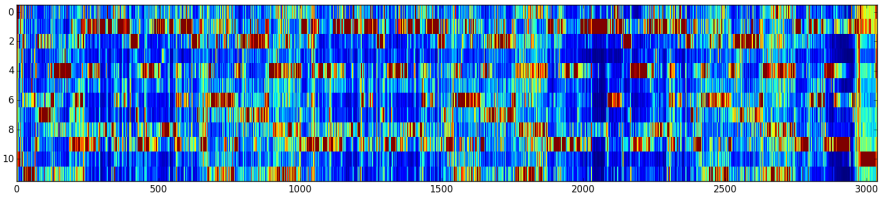


FIGURE 4.5: Example of a chromagram, generated for "Help!" by The Beatles. The blue spots represent small amount of energy and the dark red stand for the high energy in each pitch at given time.

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) [18].

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 [19]. Mathematically, this is written as:

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}\,dt \tag{4.1}$$

where $w(t)$ is the window function, commonly a Hann window or Gaussian window bell centered around zero, and $x(t)$ is the signal to be transformed.

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 at said time. We usually consider the 12 pitches commonly existing in the western popular music folded into one single octave, but one can use multiples of 12 to take the semi-tones or even smaller differences between tones into account. We achieve that 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, and very closely related to the complex Morlet wavelet transform [20].

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

$$\delta f_k = 2^{\frac{1}{n}} * \delta f_{k-1} = \left(2^{\frac{1}{n}}\right)^k * \delta f_{\min} \tag{4.2}$$

where δf_k is the bandwidth of the kth filter, f_{min} 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 $N \times M$; matrix, a chromagram is generated, where N is the number of time frames in which the musical piece has been divided [21].

4.4 Main Melody Extraction from Polyphonic Music

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. 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, as the main focus of these algorithms is to estimate the main melody from a single source.

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 [23].

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 help 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.

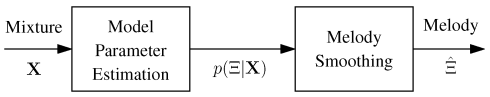


FIGURE 4.6: Outline of system proposed by Durrieu: X is the STFT of the mixture signal, $p(\Xi|X)$ the posterior probability of a given melody sequence, and $\hat{\Xi}$ the desired smooth melody sequence [22].

In their paper from 2011 [22], Durrieu together with Richard, David and Fevotte, presented an algorithm in which they assume that at any given time the music 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 aim at constraining the shapes of these STFT using temporal and spectral constraints.

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

The first one uses the source/filter Gaussian scaled mixture model (GSMM).

In short, a Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. We can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. For description of k-means clustering, we refer to Section 5.3.4.

In this GSMM model the source element refers to the excitation of the vocal folds. This is why it is linked to the fundamental frequency of the sound f_0. On the other hand, the filter part is characteristic of the vocal tract shape. This space of possibilities is then discretised so that one possible filter frequency response is considered. It 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 keeps 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.

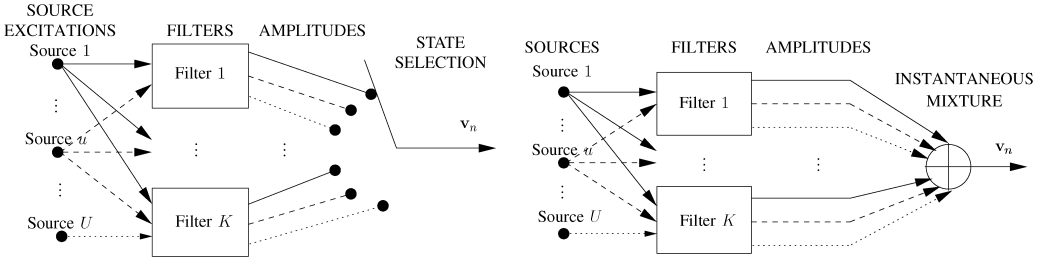
Then the likelihood of the background music is calculated.

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 $M(f,t)$ 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 calculate 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 [24].

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



(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 [22].

4.4.2 Salience Based Approaches

This type of 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.

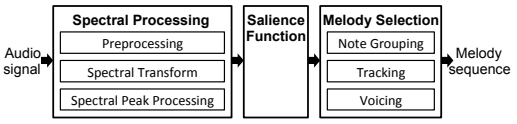


FIGURE 4.8: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creation and melody selection [23].

Usually in salience based approaches, as a first step some sort of pre-processing is applied to the audio signal to enhance the frequency content where one expects 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 a 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:

$$X_l(k) = \sum_{n=0}^{M-1} w(n) \times x(n + lH) e^{-j\frac{2\pi}{N}kn} \tag{4.3}$$

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, the authors 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 salience function provides an estimation of the predominance of different fundamental frequencies (or pitch classes) in the audio signal at every time frame. 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.

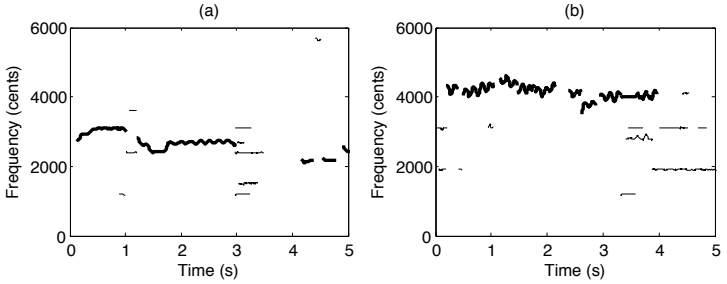


FIGURE 4.9: Pitch contours generated from excerpts of (a) vocal jazz and (b) opera. Melody contours are highlighted in bold [3].

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

After this step, the peaks of the salience function at each frame are potential f_0s 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). Then the peaks are grouped into pitch contours - time and pitch continuous sequences of salience peaks, as shown in Figure 4.9.

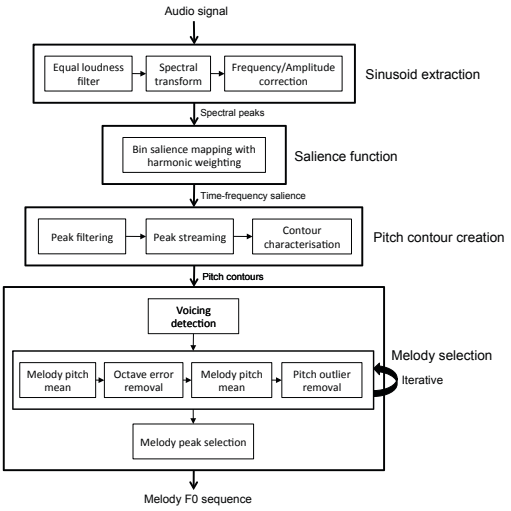


FIGURE 4.10: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creation and melody selection [3].

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, that will later guide the selection process. The 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. The first one, the voicing detection, is tasked with determining when the main melody is present.

To filter out some of the contours that are present when the main melody is absent, 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, they manage to 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 $\sigma_{\overline{s}}$:

$$\tau_v = C_s - v \times \sigma_{\overline{s}} \tag{4.4}$$

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.

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. [25]

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 sub-multiple 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 contours neighbouring 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 [23], 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:

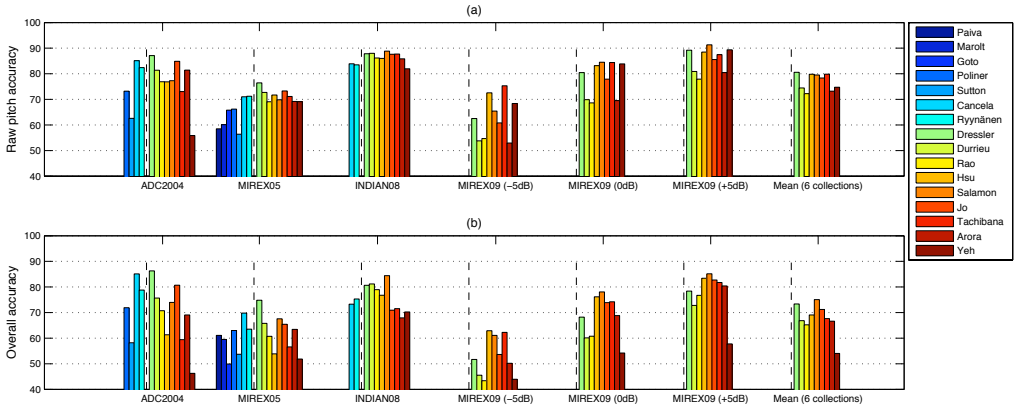


FIGURE 4.11: a) Raw pitch accuracy and b) overall accuracy obtained in MIREX by 16 melody extraction algorithms evaluated in [23]. The vertical dashed line separates the algorithms that were only evaluated on some collections (left of the line) from those evaluated on all six collections (right of the line) [23].

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 (fundamental frequency estimated by the algorithm) 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_τ 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 universally 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. Overall we can see that the average pitch accuracy over a collection lies between 70-80%.

On the other hand, when it comes to overall accuracy, the performance is lower compared to the raw pitch accuracy for all algorithms due to voicing detection being factored into the results. The importance of performance 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 could be better fitted to the purpose of this project. It outperforms the algorithm created by Durrieu, as seen in Figure 4.11, however, this is based only on the data used by MIREX to evaluate the algorithms, which might not align with our needs. In addition to this, as we are developing a game, performance of the algorithm is of crucial importance and even if one algorithm is performing slightly better than the other in terms of accuracy, we would still have to consider the time it would require the user to wait for their playable song to be generated.

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 [26]. 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 "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, neural networks can tackle problems not easy to solve by a simple computer running sequential code, 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

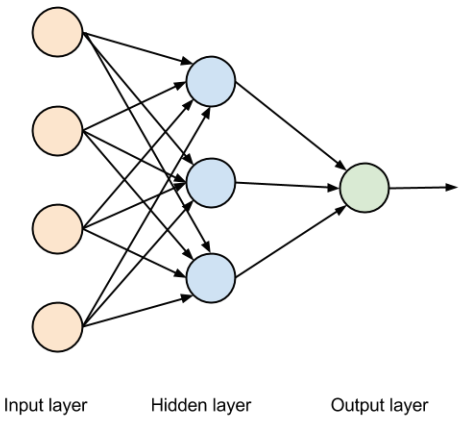


FIGURE 4.12: Diagram of a simple neural network with 4 input nodes, 3 nodes in a hidden layer and one output node.

The computational systems we usually 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. Instead, the 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 them to create a neural network and the number of neurons 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 modifying it allows us to adjust our network to improve the training outcome. The weights in most neural nets can be both positive and negative, 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 [27].

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 can 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 where the "art" of neural networking occurs.

Another part of the designer's creativity governs the rules of training. There are many 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 backpropagation, 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 [28].

Unfortunately, life is filled with situations where we are unable to extract and provide useful and valid training sets. Some of these situations could involve military action where new combat techniques and new weapons might be encountered. This unexpectedness of life and our desire to be prepared for every eventuality, there continues to be research into this field, bringing hope for future discoveries.

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 (multiplication by the weights) is applied and the output calculated. This stage is 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 connected to 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. This is depicted in Figure 4.13.

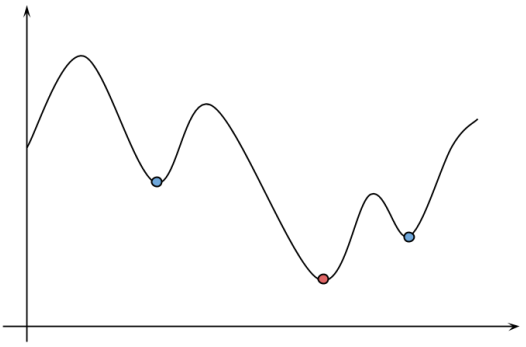


FIGURE 4.13: Diagram depicting the problem of the local minima. The x axis represents the weights and the y axis stands for the value of the error function per epoch. As the algorithm is greedy, once it hits the first minimum it will fail to reach the global minimum (red) due to the increase in the error it would have to experience first.

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:

$$W+ = W + Currentchange + (Changeonpreviousiteration * constant) \tag{4.5}$$

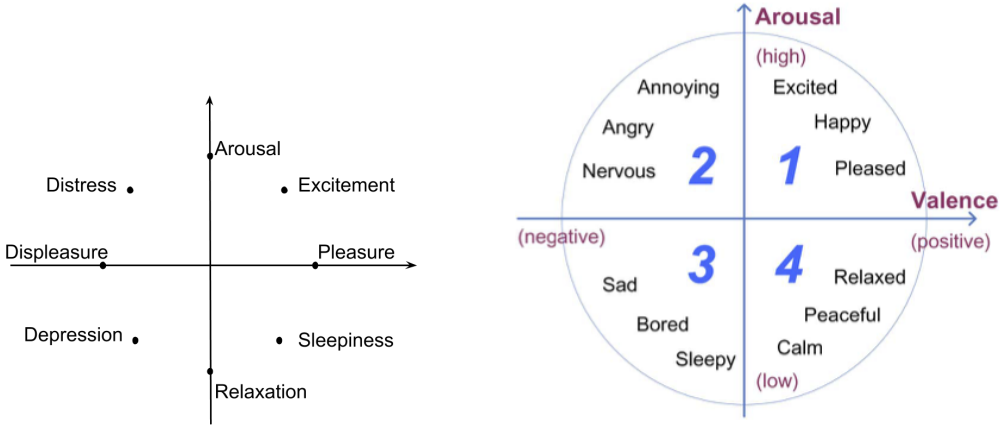
4.6 Mood Detection

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 [29], 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,



(A) Model proposed by J. A. Russell.

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

FIGURE 4.14: Diagram of both models for representing emotions.

distress, displeasure, depression, sleepiness and relaxation. Depiction of the model is presented in Figure 4.14a.

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 [30]. In the model, the amount of arousal and valence is measured along the vertical and horizontal axis, respectively.

Diagram 4.14b 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

Thanks to music's ability to affect people's emotions, much research went into discovering which feature was responsible for this. As there is no measure of "mood" alone, scientists tried different approaches in defining the emotions and their cause.

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 [31]. 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 [32]. They employed Computational Media Aesthetics, that is, analysis of two music dimensions to detect mood for music information retrieval tasks. The two dimensions – tempo and articulation, were extracted from the audio signal and mapped to one of four emotional categories; happiness, sadness, anger, and fear. After that,

they calculated a feature called relative tempo. Once the mean and standard deviation of the feature called average silence ratio in the computational articulation model was calculated, a simple backpropagation neural network classifier was trained to detect mood.

Different approach was applied by Kim and André [31]. To collect physiological data set, they used musical induction which led people participating in the research to real emotional states. This collection process was run over many weeks to exclude external emotional impact. Then the researches used four-channel biosensors to measure electromyogram, electrocardiogram, skin conductivity, and respiration changes. From that, they retrieved and analysed a wide range of physiological features to find the best emotion-relevant ones and correlated them with emotional states. Finally, the researchers performed the classification of four musical emotions (positive/high arousal, negative/high arousal, negative/low arousal, and positive/low arousal) 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 formulated 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 adopted Support Vector Regression (SVR) as a classifier using a number of chosen features.

Potentially, the treating mood recognition as a regression problem 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.

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.

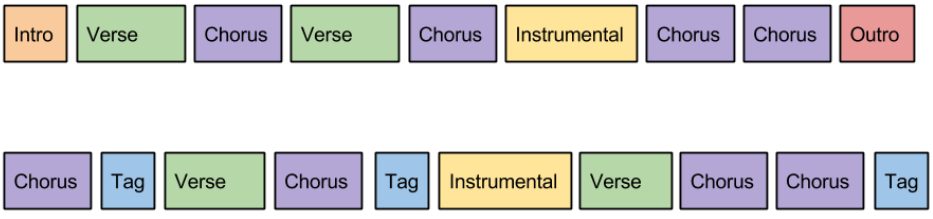


FIGURE 4.15: An example diagram of two different song structures.

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, including "Barbara Allen", "Erie Canal", and "Michael Row the Boat Ashore" [33].

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" [34].

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) [34].

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 two vectors of attributes, p and q, in an n-dimensional real vector space, the Euclidean distance between them is calculated as:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} \tag{4.6}$$

Cosine – for two vectors of attributes, p and q, in an n-dimensional real vector space, the cosine distance is represented using a dot product and magnitude as:

$$d(p,q) = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} ([_i)^2} \times \sqrt{\sum_{i=1}^{n} (q_i)^2}} \tag{4.7}$$

Manhattan – for two vectors of attributes, p and q, in an n-dimensional real vector space with fixed Cartesian coordinate system, the Manhattan distance is represented as the sum of the lengths of the projections of the line segment between the points onto the coordinate axes:

$$d(p,q) = \|p - q\|_1 = \sum_{i=1}^{n} |p_i - q_i|, \tag{4.8}$$

Correlation – for two vectors of attributes, p and q, in an n-dimensional real vector space, the correlation distance is defined as:

$$d(p,q) = \frac{(p - \mu_p) \cdot (q - \mu_q)}{||p - \mu_p||_2 ||q - \mu_q||_2} \tag{4.9}$$

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.

For instance, Foote [35] proposed the use of a self-similarity matrix (SSM for short) to visualise similarities between segments of music signals. In their algorithm, given a song, each element of a SSM represents the pairwise similarity between two respective temporal windows of acoustic features. Furthermore, Foote and Cooper [36] 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. To achieve that, a Gaussian-tapered "checkerboard" kernel is correlated along the main diagonal of the SSM. Peaks in the correlation indicate locally novel audio.

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.

Another approach seen in literature is a variant of the Nearest Neighbour Search (NSS), also known as similarity search or closest point search. NSS is an optimisation problem for finding the 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 NSS 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 [37]. 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. As the problem is not exactly solvable in general as the problem has been shown to generalize the k-means clustering problem (which is known to be NP-complete). That is why it is commonly approximated numerically [38].

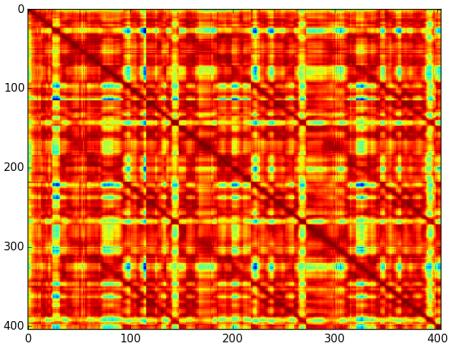


FIGURE 4.16: Self similarity matrix computed using MFCC feature for "Help" by The Beatles.

Nieto and Jehan [39] 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 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. Once they obtained the decomposition matrices, Nieto and Jehan efficiently extracted music boundaries by clustering the decomposition matrices, which take into account the repeated parts across the song instead of just detecting sudden local changes.

Some researchers joined their methods for 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 variables is generated by a sequence of internal hidden state Z. The hidden states cannot 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.

Another approach was presented by Peeters, La Burthe, and Rodet. In their paper [40], they 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). As we mentioned earlier, 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 [41] 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 Gameplay Generation

It is not really surprising that there is no current literature on the problem of automatically mapping melody to a series of buttons given an arbitrary piece of music. Most of the games available on the market are released with predetermined sets of songs, with sequences of buttons manually designed by the creators. This approach enables the developers to ensure high quality of the gameplay but limits the potential experience and discourages people whose music taste differs from fans of the most popular and generic music.

Any other releases or fan-made patches that enable the users to play to a wider range of music, focus on rhythmic representation of the song, disregarding the fundamental frequency of the main melody.

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. Such algorithm should produce a consistent output that makes the gameplay as intuitive for the user as possible.

Chapter 5

Design and Implementation

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

First, we describe our solution to the mood detection problem. We attempt that by trying to determine which musical features are the most correlated to the AV values of the music's emotion. Having found the set of features that have the biggest impact on our perception of the emotions in the song, we train a neural network with data containing chosen features. This way we create a way of determining the arousal and valence values of any musical track. The system is later used in the implementation of our game. In addition to this, by investigating the impact of different parameters, we make sure out network has as good performance and accuracy as possible.

Next, we 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 evaluate performance of both of them on data from recent pop culture to determine their performance and fitness in this project. In addition to this, we employ post processing to further modify the output of the chosen algorithm and smooth out the pitch contours, removing outliers that could interfere with the gameplay. Thanks to parametrising and applying the sliding median we are able to produce pitch coutours of different difficulty to play.

The next section describes our attempt to design and develop an automated music segmentation system. We investigate two different features – Mel-frequency cepstral coefficients and harmonic pitch class profiles, to generate a self-similarity matrix and apply the convex non-negative matrix factorisation (C-NMF) to retrieve its decomposition matrices. Using those matrices, we then investigate two different ways of finding boundaries of a musical track. Finally, we describe our process of inventing a novel way of labelling segments of the song.

Last, but not least, we talk about the game itself, its architecture, flow of use and design choices made. We describe the algorithm for generating buttons that constitute the main part of the gameplay. We also detail the way we tie in the music analysis performed into the game to make the most of it.

5.1 Mood Detection

A common reason for engaging in listening to music is that it 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 common points in perceived emotion across different listeners based on the characteristics of the music [42]. 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 [30].

As we described in Section 4.6.1, there is a relation between valence and arousal values for a musical track and the mood 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 [43], we implemented an extractor to retrieve certain features from a song. We chose the features on the impact we expected them to have on the perceived mood of a musical piece:

average loudness - describes dynamic range (the ratio between the largest and smallest possible values of the signal) of loundness. The values are rescaled into the [0, 1] interval on a per window basis - if 0, it means that the signal exhibits a 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 or tracks that vary greatly in loudness.

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 or more peaceful the track seems to be, implying the low arousal value. When examining multiple musical tracks, we noticed that the happier or angrier songs can also have such silent gaps, but they tend to be much shorter and less frequent.

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 "centre 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, and formalise it acoustically as an indication of the amount of high-frequency content in a sound [44]. 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. In audio processing, it 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 E_s 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 [45], Ferguson, Kenny and 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 as loud beats can be found in parts of song with higher energy.

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 [46]. 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.

mean ZCR (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 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:

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

where s is a signal of length T and the indicator function $\mathbb{I}\{A\}$ yields 1 if A is true, 0 otherwise.

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.

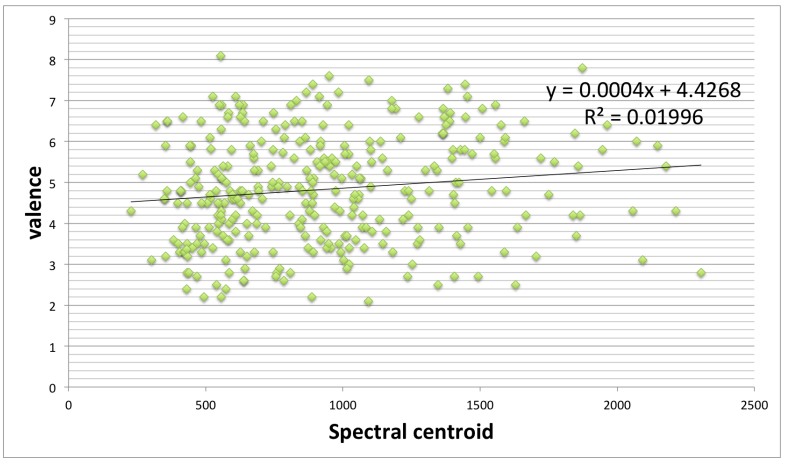


FIGURE 5.1: A graph representing a correlation between spectral centroid and valence values.

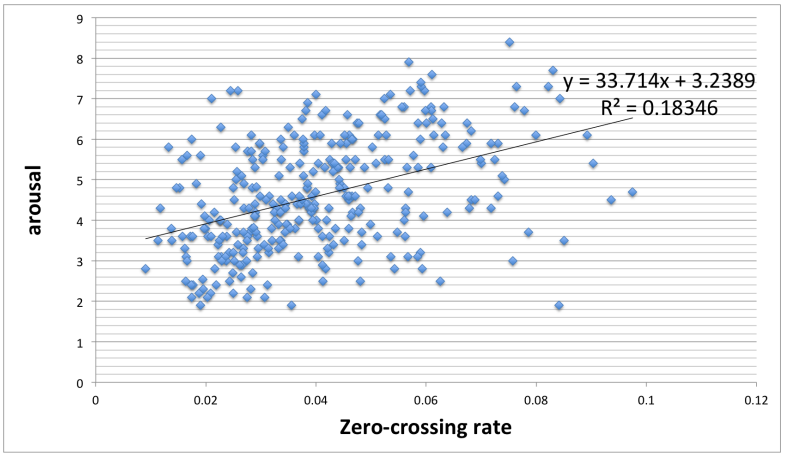


FIGURE 5.2: A graph representing a correlation between zero-crossing rate and arousal values.

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 [47], 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 Figured 5.1 and 5.2, the rest are included in Appendix A, Section **??**) 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

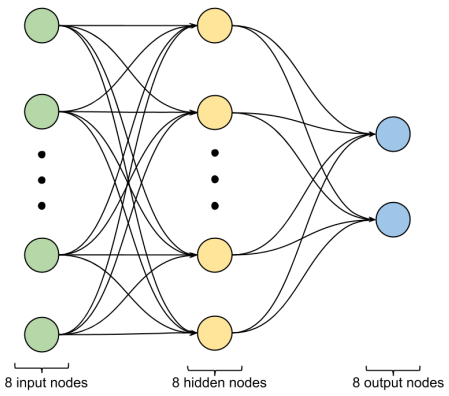


FIGURE 5.3: A diagram depicting the structure of our artificial neural network for mood detection.

Our goal was to train the network to predict mean participant valence and arousal values for musical excerpts.

Our 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, loudness, 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 8 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 0 to 10 inclusive, the network outputs were rescaled back. The training set consisted of 50 input and output arrays. 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 [48] Python library for neural network implementation.

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

No. of Nodes	RMSE 1	RMSE 2	result 1	result 2
1	0.0727638	0.0740583	0.0998089	0.0978822
2	0.0717966	0.0709793	0.1130468	0.1124054
3	0.0722213	0.0733605	0.1554125	0.0948397
4	0.0702014	0.0699922	0.1026024	0.1153731
5	0.0659433	0.0693361	0.1165588	0.1042320
6	0.0722427	0.0751383	0.1422484	0.1188432
7	0.0698701	0.0678483	0.0889542	0.1035373
8	0.0692459	0.0664240	0.1314129	0.1360981
9	0.0676911	0.0707275	0.1281395	0.1042317
10	0.0684399	0.0673887	0.1405055	0.1751565
15	0.0671656	0.0673142	0.1165631	0.1392658
20	0.0737978	0.0720621	0.1604241	0.1362109
50	0.0669456	0.0694139	0.1641328	0.1716036

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

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.

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.

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

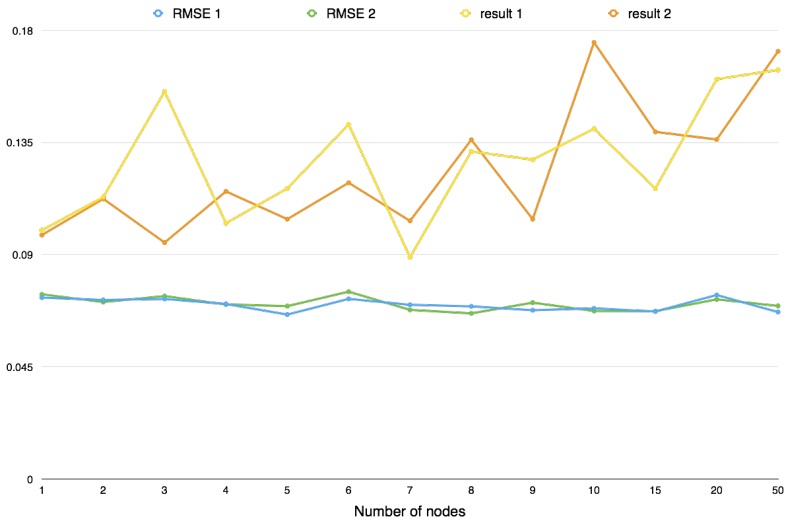


FIGURE 5.4: Data presented in Table 5.1, plotted on a diagram.

Learning Rate	RMSE	result RMSE
0.3	0.070797	0.145788
0.25	0.069934	0.163193
0.2	0.066799	0.155219
0.15	0.072422	0.104971
0.1	0.068426	0.100719
0.05	0.069596	0.097935
0.01	0.068946	0.090954
0.005	0.072402	0.130734
0.001	0.079665	0.112620

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

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. Learning rate is essentially a training parameter that controls the size of weight and bias 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 5.3.

5.2 Main Melody Retrieval

As mentioned in Section 4.4, we looked into two different algorithms for main melody extraction. The first one, implementing the source separation based approach, was designed by Durrieu [22], and the other one, using the salience approach, by Salamon and Gómez [3]. In this section, we evaluate the performance of each of the algorithms in light of usefulness in our project. Furthermore, we present a way of postprocessing of the chosen algorithm to further smooth out the estimated pitches of the main melody estimated.

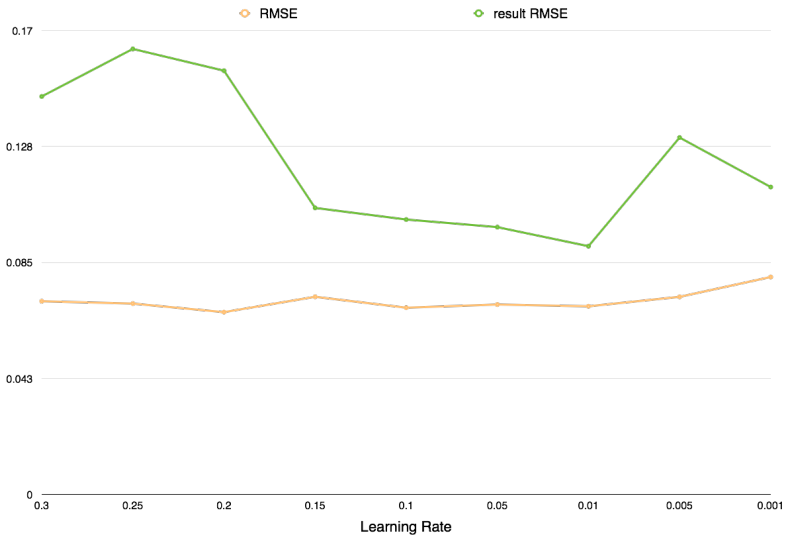


FIGURE 5.5: Data presented in Table 5.2, plotted on a diagram.

Title	Durrieu Recall	Durrieu False Alarm	Salamon Recall	Salamon False Alarm
Imagine	0.996	0946.	0.604	0.490
Johnny B. Goode	0.999	0.962	0.834	0.528
Like a Rolling Stone	0.993	0.873	0.634	0.581
Respect	0.959	0.494	0.652	0.198
Satisfaction	0.997	0.948	0.778	0.583

TABLE 5.3: Table showing the recall and false alarm for each of the algorithms when analysing the against ground truth files.

5.2.1 Accuracy

Although both of the algorithms were evaluated at some point at the MIREX conference, the data they were tested against does not fully align with our needs. For this purpose, we ran both melody extraction methods against the most popular songs of different genres commonly considered the best [49].

To generate the ground truth, we used MIDI versions of files, created with separable channels so that the main melody is easily extracted. As a first step, we computed the voicing recall (the fraction of voiced frames in the ground truth reference indicated as voiced in the estimation) and false alarm rates (the fraction of unvoiced frames in the ground truth reference indicated as voiced in the estimation). Our findings are shown in Table 5.3.

As we can see from the Table 5.3, although initially the recall rate of Durrieu's implementation looked promising, the rate of the false alarms makes us think that the algorithm, in fact, does not handle the filtering out of the unvoiced frames too well and detects the f_0 of the main melody forcefully. In total, Durrieu exhibited an average of

Title	Durrieu Pitch	Durrieu Chroma	Salamon Pitch	Salamon Chroma
Imagine	0.239	0.348	0.124	0.434
Johnny B. Goode	0.020	0.109	0.727	0.769
Like a Rolling Stone	0.020	0.354	0.018	0.313
Respect	0.529	0.588	0.260	0.365
Satisfaction	0.086	0.343	0.071	0.527

TABLE 5.4: Table showing the recall and false alarm for each of the algorithms when analysing against ground truth files.

98.28% recall when it comes to detecting the voiced frames but at the same time, it suffered from an average of 84.46% false positives. On the other hand, Salamon's solution did have a higher miss rate, but at the same time, its false alarm rate never exceeded 0.6. Still, we believe that an average of 70% recall with 47.6% false positive rate is an impressive achievement for Salamon.

Although the results in the Table 5.3 are very reliable, they rely on midi files that contain the main melody performed by a digital instrument and not a human voice. This means that all the advantage the algorithms may have by analysing the main melody for vocal features such as vibratos, as it is described in case of Salamon's algorithm, are underutilised.

In addition to this, we analysed the accuracy of pitch prediction for each of the algorithms. To do so, we computed the fraction of voiced frames in reference file for which the algorithms provided a correct frequency values. As we can see in Table 5.4, performance of each of the algorithms varies gravely depending on the track. The average pitch prediction for Durrieu was 17.8%, whereas Salamon exhibited an average prediction of 24%. We found those numbers a bit disappointing, however, we suspected that a big part of the error was caused by the octave error, which are not so important in our case, as we are more interested in how the melody changes and as long as there octave jumps are consistent, we do not mind the error introduced. That is why we also calculated the raw chroma accuracy, where all the pitches are mapped onto one octave. In this case, Salamon's performance increased to 48%, whereas Durrieu showed an improvement from 17.8% to 34.84%.

5.2.2 Performance

Although in scientific environment not so important, in case of implementing a game, one of the most important criterion in choosing tools is their speed. We cannot imagine any successful game forcing their users to wait long hours between choosing a music track and being able to play the generated song. The user simply cannot be expected to plan a few hours ahead when and what song they will feel like playing. That is why, apart from investigation of the accuracy of the candidate algorithms, we have to take into account their speed. We present the times it took to generate main melody

Title	Length	Durrieu	Salamon
Imagine	3m 12s	4h 5m 53s 570ms	26s 422ms
Johnny B. Goode	2m 20s	3h 2m 14s 760ms	29s 932ms
Like a Rolling Stone	1m 32s	1h 59m 52s 630ms	9s 193ms
Respect	2m 35s	3h 23m 38s 610ms	12s 597ms
Satisfaction	3m 43s	4h 47m 16s 330ms	15s 65ms

TABLE 5.5: Table showing the recall and false alarm for each of the algorithms when analysing against ground truth files.

estimated by each of the algorithms for every song in Table 5.5. As we can see, Salamon analyses the songs incomparably faster than Durrieu. In addition to this, it achieves this with higher pitch prediction rate.

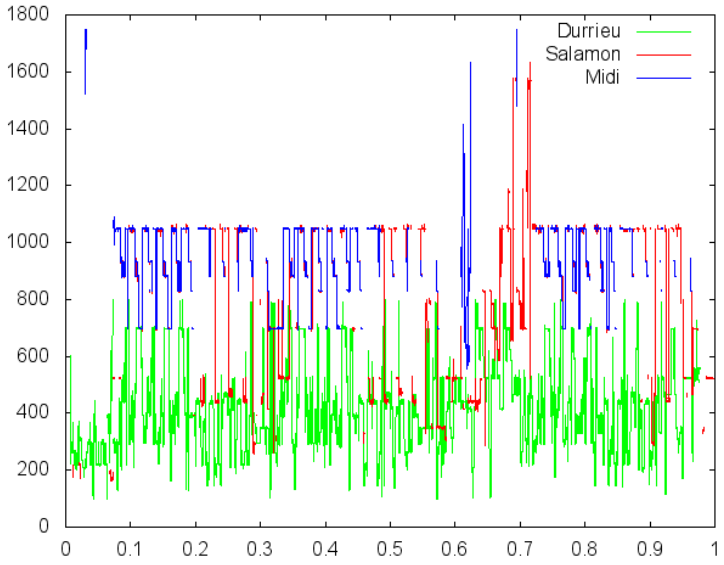


FIGURE 5.6: Pitch contours of the ground truth (blue), Durrieu (green) and Salamon (red) for "Johnny B. Goode" by Chuck Berry.

5.2.3 Postprocessing

Once we have computed the pitch estimates, we need to post process them to make them more suitable for our application – we would like to smooth out the pitch contours to avoid sudden spikes. In addition to this, it would be beneficial to be able to control the extent of the smoothing to allow variation in difficulty of the playable song that is going to be generated out of it.

Most smoothing algorithms are based on the "shift and multiply" technique, in which a group of adjacent points in the original data are multiplied point-by-point by a set of numbers (coefficients) that defines the smooth shape. The products are added up to

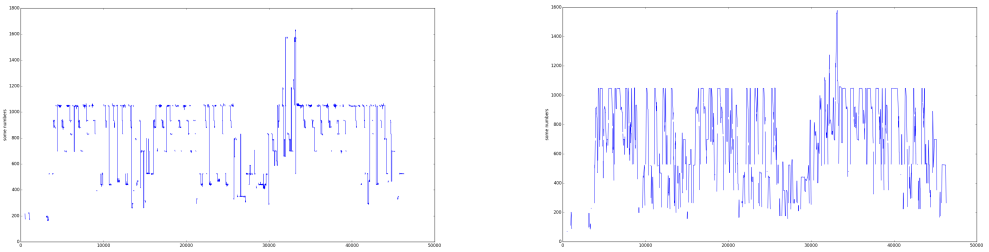


FIGURE 5.7: The pitch contours of the input (left) and created from smoothing the estimated main melody of "Johnny B. Goode" by Chuck Berry with additional introduction of false positives (left) and not (right), using the rectangular smooth with $m = 101$ (right).

become one point of smoothed data, then the set of coefficients is shifted one point down the original data and the process is repeated.

The simplest smoothing algorithm is the rectangular or unweighted sliding-average smooth; it simply replaces each point in the signal with the average of m adjacent points, where m is a positive integer called the smooth width. For example, for a 3-point smooth ($m = 3$):

$$S_j = \frac{Y_{j-1} + Y_j + Y_{j+1}}{3} \tag{5.3}$$

for $j = 2$ to $n - 1$, where S_j the jth point in the smoothed signal, Y_j the jth point in the original signal, and n is the total number of points in the signal.

The results of applying the rectangular sliding average smooth can be seen in Figure 5.7. As we can see, even for a window size of 101, which actually is not big when compared

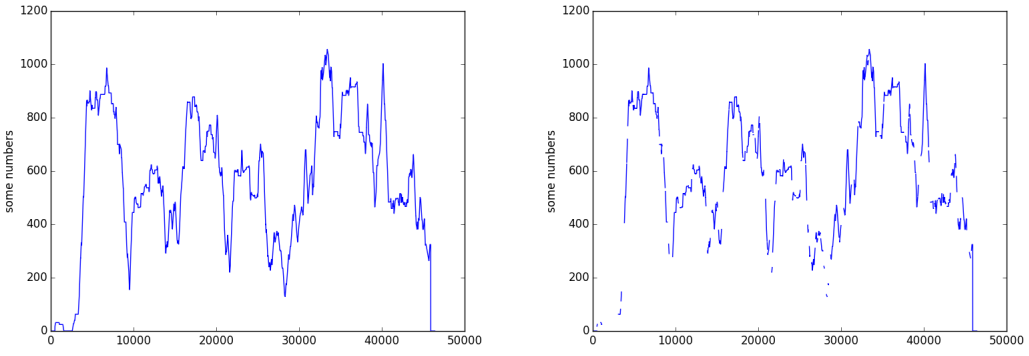


FIGURE 5.8: The pitch contours created from smoothing the estimated main melody of "Johnny B. Goode" by Chuck Berry with additional introduction of false positives (left) and not (right), using the rectangular smooth with $m = 1001$.

to the length of the whole song (in case of this example is over 50,000 frames), the zero values (so the unvoiced frames) drag down the pitch values around them. This is a desirable outcome when dealing with the outliers that were falsely detected as voiced, but introduces more error into correctly detected frames that start the note (so for instance, the first note after a pause).

Another solution to the problem is to use the sliding median filter. The main idea behind it is to run through the signal entry by entry, replacing each of them with the median of neighboring entries. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire signal.

For 1D signals, the most obvious window is just the first few preceding and following entries, whereas for 2D (or higher-dimensional) signals such as images, more complex window patterns are possible (such as "box" or "cross" patterns). Note that if the window has an odd number of entries, then the median is simple to define: it is just

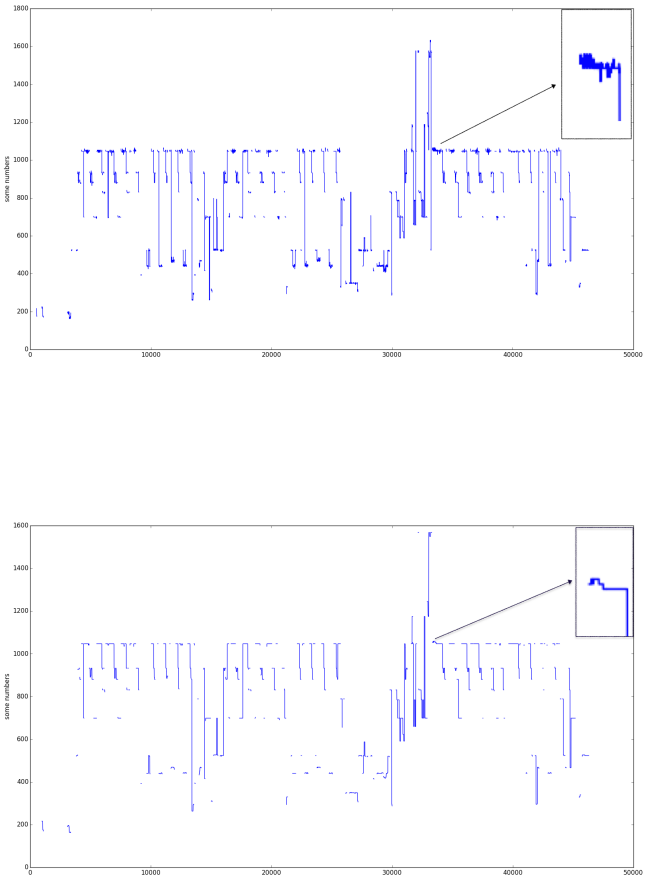


FIGURE 5.9: The pitch contours of the input (left) and created by applying the sliding median with window size $w = 100$ to the estimated main melody of "Johnny B. Goode" by Chuck Berry.

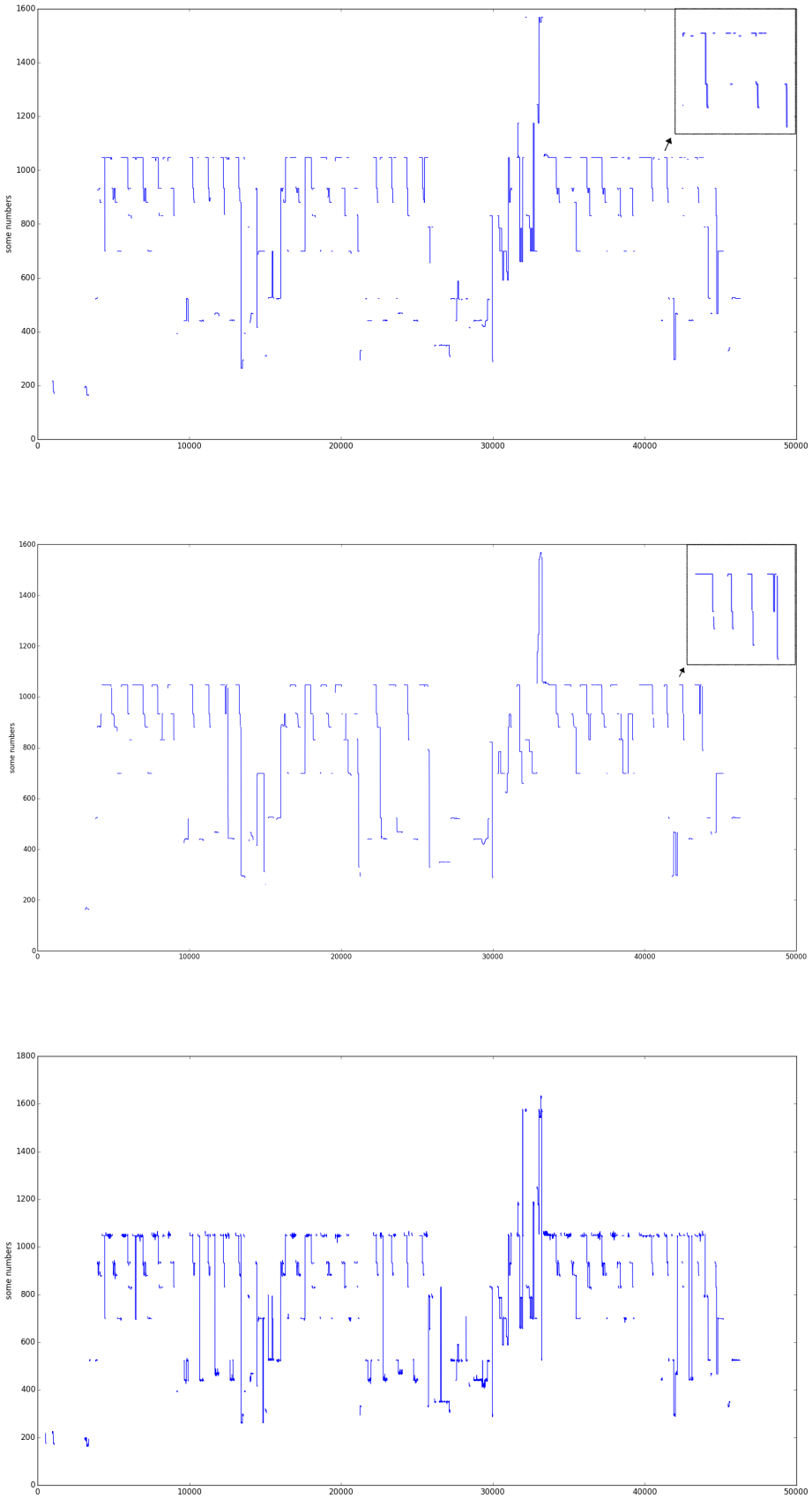


FIGURE 5.10: The pitch contours created by applying the sliding median with window size $w = 100$ (top), $w = 300$ (middle) and the initial input (bottom).

the middle value after all the entries in the window are sorted numerically. For an even number of entries, there is more than one possible median.

The impact of applying the sliding median of size 100 on the estimated pitches can be seen in Figure 5.9. On the other hand, comparison of the results for the different window sizes can be seen in Figure 5.10. As we can see, applying a bigger window size smooths the contours even more. Although this way the main melody loses some of its elements, like when the voice is quickly vibrating, it can prove useful when generating simpler songs to play, as the changes in the pitch will be less frequent, making the melody less complex and hence, less interaction will be required from the user.

An important thing in our application is that we do want to preserve our zero values in our smoothed pitch coutours. If we fail to do so, we introduce additional false positives in our voicing detection. The visualisation of the problem can be seen in Figure 5.8.

5.3 Structure Retrieval

Understanding the structure of music (i.e. localisation of different parts such as 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 needed to provide it with some data. We investigated two possible values - *Mel-frequency cepstral coefficients* and *harmonic pitch class profile*.

Harmonic pitch class profiles (HPCP) is a vector of features extracted from an audio signal, based on the Pitch Class Profile descriptor. It is an enhanced pitch distribution feature which is described by 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.
- 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.
- 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.

An alternative to using HPCPs as the features to base the algorithm on is *Mel-frequency cepstral coefficients*.

The term "cepstrum" refers to 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.

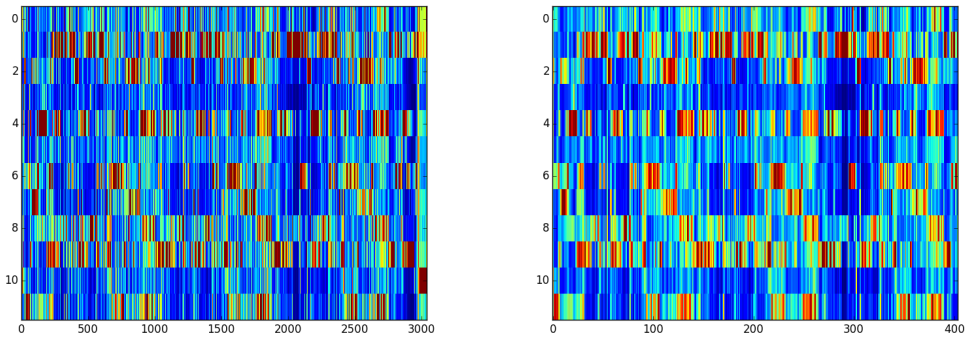
The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

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 perceptually based spectral features originally designed for speech processing applications. Similarly to the pitch decomposition, the frequency bands are positioned logarithmically on the so-called mel scale, which approximates the response of the human auditory system. In the music context, MFCCs have turned out to be useful in capturing timbral characteristics.

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 [50]

MFCCs are commonly derived as follows:



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

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

FIGURE 5.11: Harmonic pitch class profiles chroma features calculated for a song by The Beatles- "Help!".

- 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.

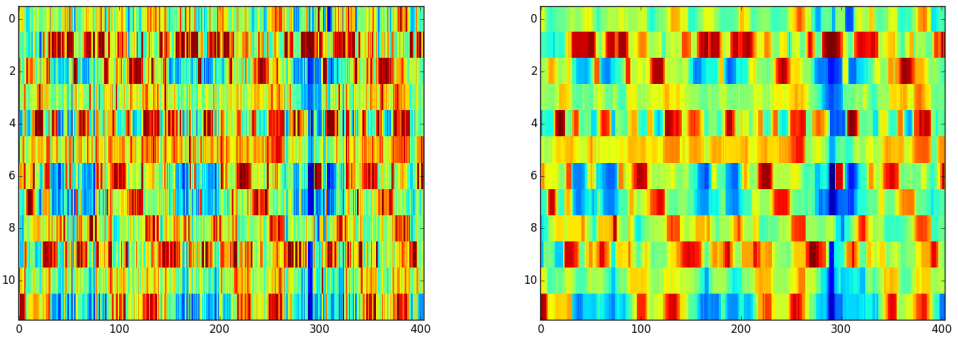
5.3.2 Feature Preparation

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 music 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 expected the opposite behaviour - good performance on parts that are different in terms of timbre.

Harmonic Pitch Class Profiles

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

First, we needed to we synchronise our data with the beats detected in the musical track. This process allowed to reduce local variation by summarising frame-wise features



(A) Chroma feature after applying log normalisation.

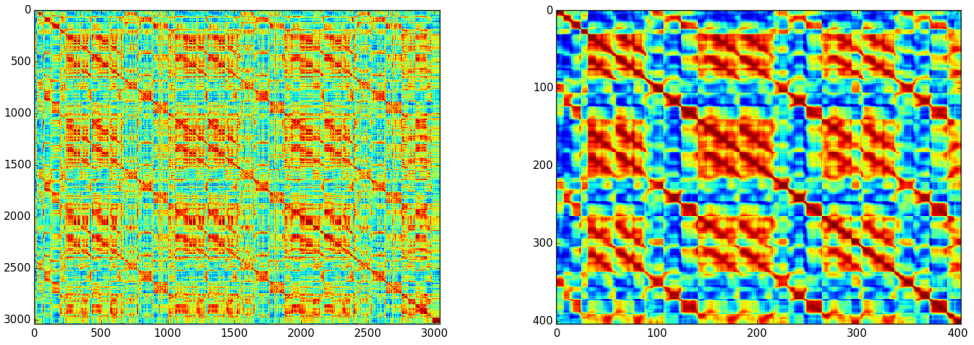
(B) Chroma feature after applying sliding median filter of size $h = 9$.

FIGURE 5.12: Beat-synchronised chroma created for song "Help!" by The Beatles with applied enhancements.

that occur between two beats, yielding fewer but longer beat-synchronous frames. The rationale for doing so was 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 managed to largely normalise variations in tempo. However, the main advantage of applying the beat-synchronisation was that we managed to reduce the amount of data to analyse, and hence, the size of the matrix, we were operating on.

This led us to beat-synchronous chromagrams. A diagram of a chromagram after beat synchronisation can be seen in Figure 5.11b. As we can see, the size has decreased dramatically, which makes the segmentation process computationally cheaper.

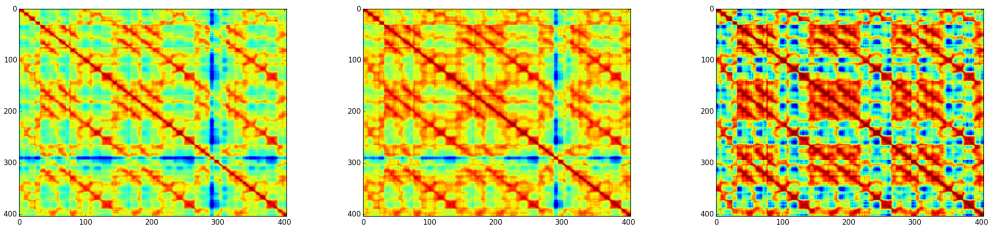
Following the beat-synchronisation, we applied log normalisation to the chroma feature. This allowed us to reduce the effect the outliers from the trend would have and further



(A) Self-similarity matrix generated from harmonic pitch class profiles chroma without further enhancement.

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

FIGURE 5.13: Comparison of SSM generated from unprocessed and enhanced chromas, using correlation distance.



(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 5.14: Comparison of SSM computed using different distance formulas. The SSM calculated using correlation distance can be seen in Figure 5.13b.

improve the contrast between the related and unrelated beat frames. The enhancement achieved by applying log normalisation can be seen in Figure 5.12a.

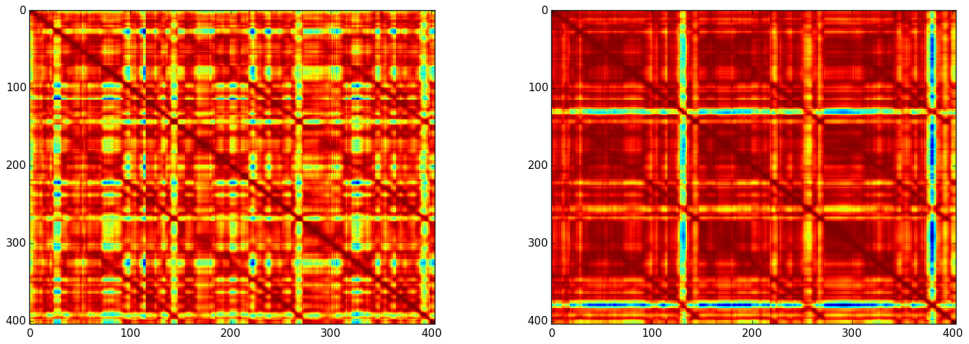
As the next step, we applied a sliding median filter of size h is run against each of the beat-synchronous and log-normalised chromagram channels. Thanks to the median filter, we could come up with sharper edges than with a regular mean filter. This became really useful in obtaining section boundary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

By filtering features across time, we retained the most prominent chromas within the h-size window and removed smaller artefacts, which are irrelevant in our context. The Figure 5.12b presents the chromagram after applying the sliding median filter.

We then proceeded 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 i and j. The result of every such comparison is stored in a $N \times N$ symmetric matrix D, such that $D(i, j)$ contains said distance. In particular, $D(i, j)$ stores the same value as $D(j, i)$, 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 5.14. As we can see in Figures 5.14a and 5.14b, the contrast achieved for SSM produced using Euclidean and Manhattan distances was much weaker. Not only there were fewer blue spots signifying small or no similarity between points, but the amount of points that were significantly similar was also reduced.

When we look at the SSM computed using cosine distance, we can notice that the amount of the similar points increased, more similar to the the one generated using the correlation distance. However, the correlation distance on Figure 5.13b contains more dark blue spots, implying that it exposes more beats that are, in fact, not similar. This is why, in our design of the structure retrieval of a song we decided to use SSM computed using correlation distance.



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

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

FIGURE 5.15: Comparison of SSM generated from unprocessed and enhanced MFCCs, using correlation distance.

Mel-frequency Cepstral Coefficients

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

Next, we had to determine whether log normalisation improved the clarity of the SSM. As we can see in Figure 5.15, the use of log normalisation could decrease the amount of segments found, as more similar points are exposed.

Finally, we computed 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 (**??**). 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 5.15b.

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.

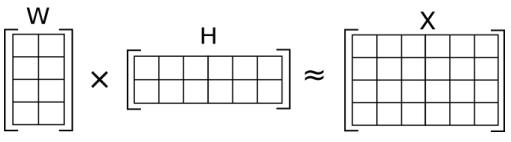


FIGURE 5.16: Illustration of approximate non-negative matrix factorization: the matrix X is represented by the two smaller matrices W and H.

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 constraint 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:

$$\boldsymbol{w}_j^T = \boldsymbol{x}_1^T f_{1j} + \boldsymbol{x}_2^T f_{2j} + \ldots + \boldsymbol{x}_N^T f_{Nj} \qquad j \in [1:k] \tag{5.4}$$

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: $X \approx XFH$.

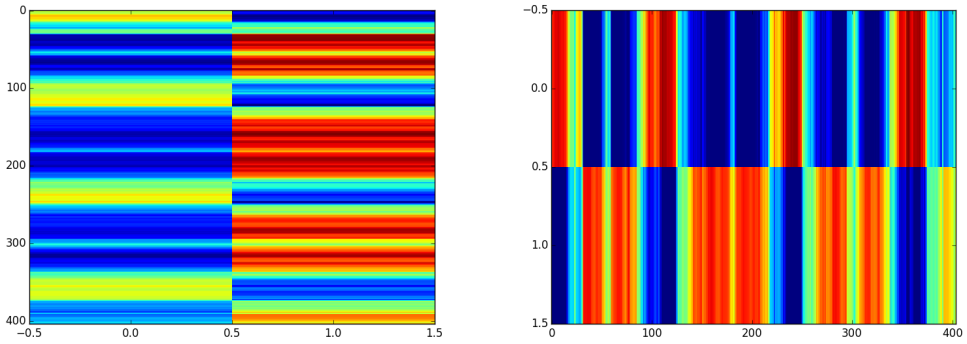
In C-NMF, the matrix W is a set of convex combinations of the rows of the input matrix X, which contrasts with NMF, where no such constraint exists. This means that, each row x_i represents similarity of the time frame i with the rest of the time frames, storing information about the time frame i across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

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

Apart from this, another important benefit of C-NMF over NMF is that matrices W and H become naturally sparse when adding the convex constrain. In case of NMF the H 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 [39].

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 5.17.



(A) The cluster matrix W. (B) The activation matrix H.

FIGURE 5.17: The result of C-NMF computed for "Help!" by The Beatles with rank $k = 3$.

Pure C-NMF Approach

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.

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 quite simple, produced a neat and quite accurate output, with a few, but meaningful segments. In the next section we describe an attempt of improving our results that did not yield results we hoped for, and hence was left out in the final implementation.

K-means Clustering

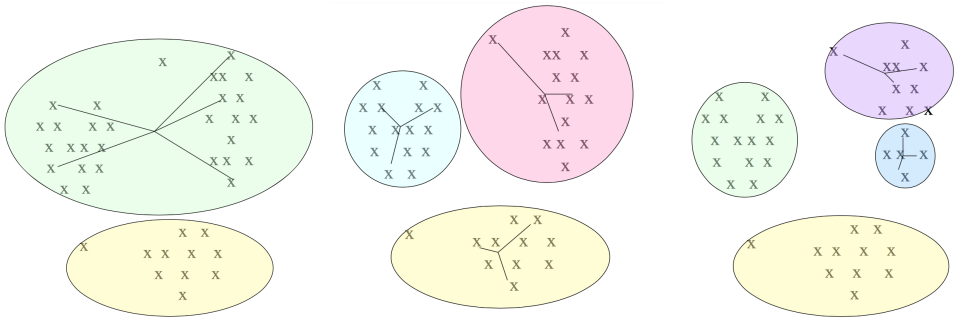
K-means clustering is considered one of the simplest unsupervised learning algorithms that can solve the well know clustering problem. It follows a simple and easy way of classifying a given data set through a certain number of clusters (assume k clusters).

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.



(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 5.18: Diagrams depicting impact of the k value on the clustering result [51].

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, as described in Section 5.3.4. 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.

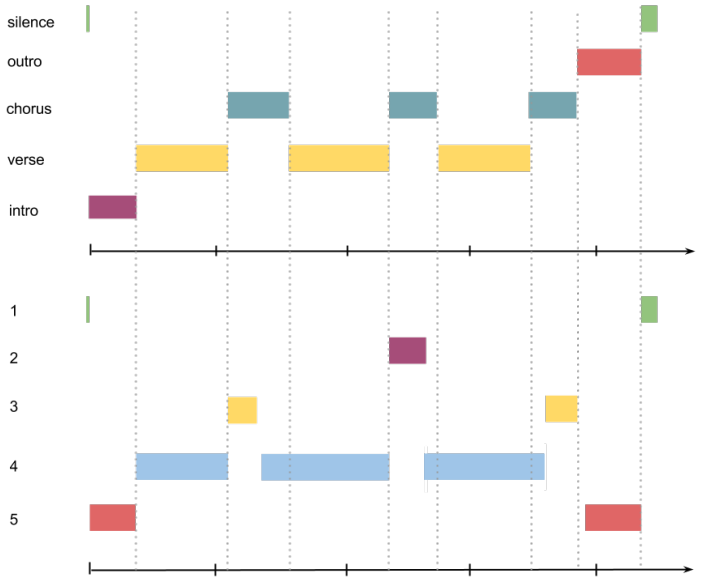


FIGURE 5.19: A depiction of the pure vocals detection as a labelling method (bottom) compared with manual segmentation and labelling (top).

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.

A visualisation of our segmentation and labelling using only vocal/instrumental heuristics which can be seen on Figure 5.20. The diagram makes it evident that our boundary finding works quite well. As we can see, each beginning of the chorus is detected with

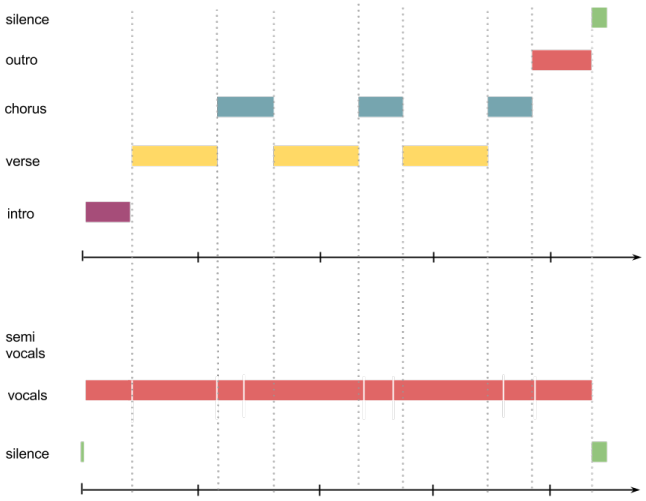


FIGURE 5.20: A depiction of the pure vocals detection as a labelling method (bottom) compared with manual segmentation and labelling (top).

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.

However, this way alone, we lose the distinction between the vocal parts, for instance, between the verse and the chorus. We could implement a system of heuristics to assign labels like chorus or verse, but unfortunately there is no straightforward way of doing so as the song structures differ greatly across the popular music. For instance, many songs drop the verse-chorus structure to start with a chorus, like "She Loves You" by The Beatles.

Our final approach was to merge the two approaches mentioned earlier into one algorithm. Ideally, we wanted to preserve the variation in the vocal parts, as it was presented in the first approach where multiple parts were classified with the same class, while being able to tell when the main melody of is present, and when it is not. Moverover, we had to find a solution to the problem of determining which segments contain chorus, which ones represent the verse and which should be classified otherwise.

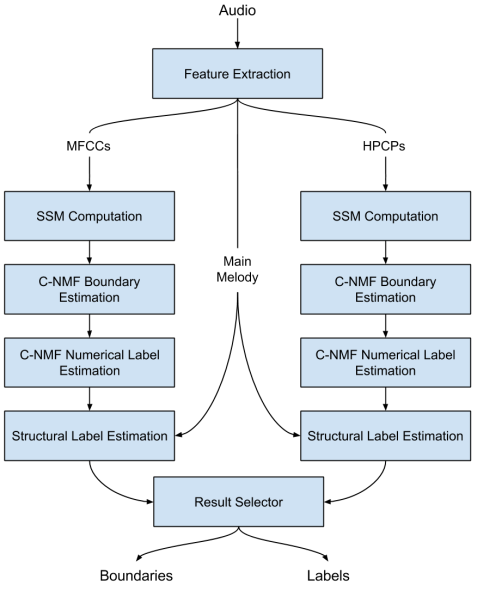


FIGURE 5.21: Depiction of the final design of our structure retrieval system.

To make it possible, we first employed our

initial labelling algorithm to retrieve numerical labels for each of the detected segments. This way we were able to find relationships between different segments. Once we know which segments bear high similarity, we average the amount of percentage of voiced frames per segment. Having done that, we look at the two most repeated segments. The one with the higher average of the voiced frames is labelled as a chorus, and the other one is determined to be a verse. The first segment is labelled an intro if it bears not similarity to other segments. If it is classified the same way the last segment is, we label both if them as "intro/outro". If our algorithm finds another segment similar to any of them, they are marked as the bridge.

5.3.6 Final Design

Once we have designed the the labelling algorithm, we researched the impact of the choice of feature on the performance of the segmentation. We had noticed that some songs achieved better results when HPCPs were chosen to lead the computation and retrieve the song segmentation whereas the others performed better with MFCCs taken into account.

To solve the dilemma of the choice of feature, we implemented a selector, which inspecting the outputs produced using HPCPs and MFCCs, selects the one which is most likely to reflect the structure of the song. It investigates boundaries and labels returned by processing each of the SSMs and looks at their structure - the length of the silences, length amount of segments detected and how the segments fit into their neighbourhood. Based on those observations, it ranks both outputs based on their likelihood and selects the most likely one.

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 [52].

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.

For the music analysis part of the project we used Python. This is because there are many libraries available in this language for both training of artificial neural networks and music analysis.

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. [53].

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 song.

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 songs 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

Success of any project or product depends on many factors - the performance, design, innovation and design. Each of the elements it consists of contribute to at least one of them. In this chapter we evaluate our project in attempt to assess its success.

First, we focus on quantitative analysis. We present the results of testing and validating of our neural network for music emotion prediction using fresh and unseen data and contrast them with existing literature. Then we move on to evaluation of the boundary detection system by comparing the performance and accuracy it yields with two other algorithms. Finally, we examine the results of the labelling algorithm.

Last, but not least, we move on to user experience research. We go over user testing and its outcome, as well as how it was used to further improve our game.

6.1 Evaluation of Mood Detection system

We designed our neural networks to as means of calculating valence and arousal ratings of songs using audio features we extract.

To evaluate its performance we have to have a way of telling how successful it is in guessing the values. To achieve this, we activated the network on 21 unseen music tracks that we knew valence and arousal values of and gathered the output network produced for it. Then, we computed the root mean square error between the ground truth ratings and network-predicted outputs across all segments of all the test melodies. The network's performance total RMSE was 0.08895 on scale from 0 to 1 or 9%. The plot of the expected and predicted values can be seen in Figure 6.2.

In contrast, in their paper [42] Vempala and Russo trained a neural network with 1.14 error on scale 1 to 9, or 14.3%. The better performance could be caused by the more suited choice of features or simply because we trained our network on a bigger data set.

If we look at the two predicted values separately, the coefficient of determination for the arousal reached 59.24% and 40.34% for valence for our network.

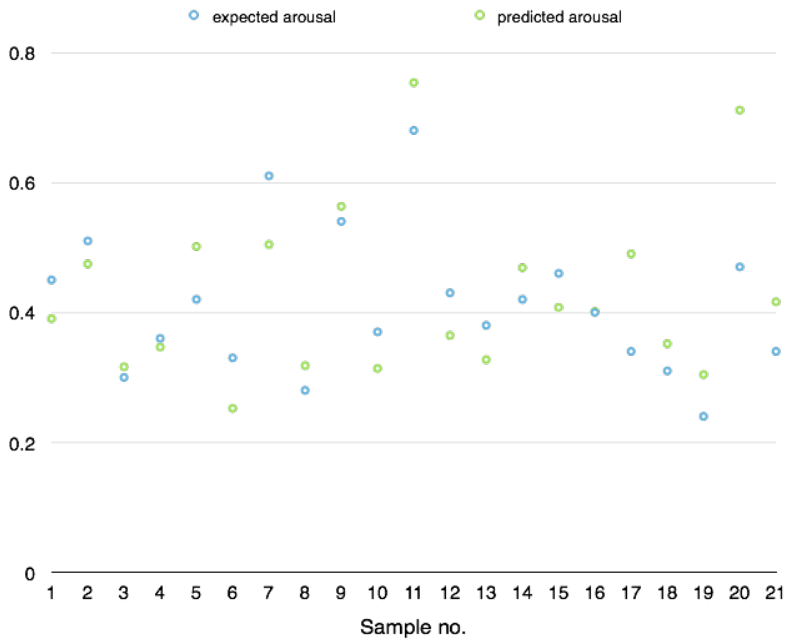


FIGURE 6.1: A plot of the expected and predicted .

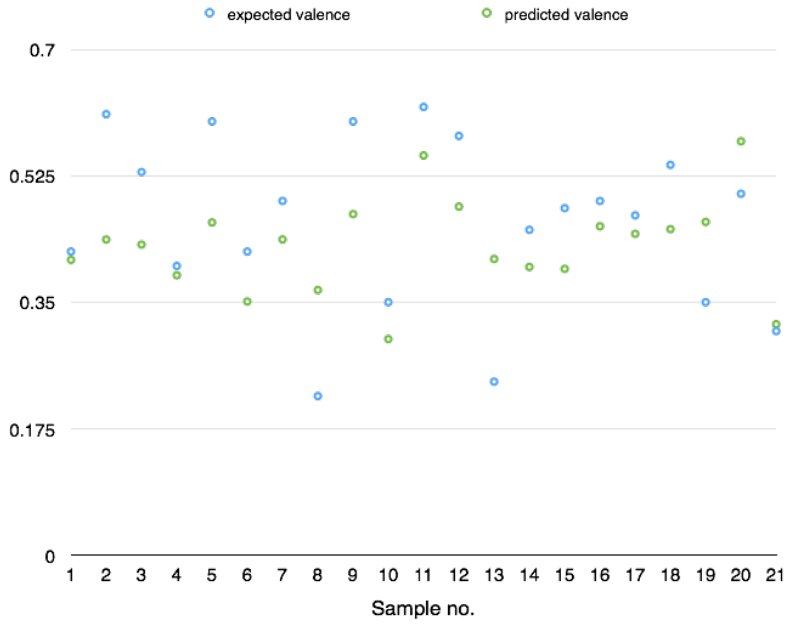


FIGURE 6.2: A plot of the expected and predicted valence.

In comparison, Yang and Lin [4] achieved a lower success rate, as their R^2 scored 58.3% for arousal and 28.1% for valence. This might be because of a different set of features chosen or the neural network was simply better for calculating the relationship between the set of features and the mood value of the song.

The RMSE our network exhibited was 9%/ This means that the overall accuracy of the network can be estimated at the level of 91%. This is a result comparable even to classification solutions to the problem, for example proposed by Liu, Yang, Wu, Chen [**?**], where at the overall performance the network reached 78.97%.

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.

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.

6.2 Boundary Detection

The first towards the evaluation of the segmentation algorithm was to gather the ground truth data to test against. To remove bias caused by personal preferences in music, we sought external sources for help in selection of the tracks. This is why we consulted the top charts created by music magazines and online portals such as Rolling Stone, Billboard, Gibson etc. The full list can be found in [49].

Once we have retrieved the lists with the most popular songs across genres, we selected one song per top list based on our familiarity. Although this might introduced some sort of personal preference into the dataset, it also allowed us to more intuitively create the ground truth data, as the familiarity with a song makes the manual labelling more obvious and precise. For instance, in “Blitzkrieg Bop” by The Ramones, the bridge is

Song	Ground	Foote	Kaiser	Ours
The Number of The Beast	17	9	15	20
Rock With You	11	9	15	17
Blitzkrieg Bop	14	9	14	14
This Land Is Your Land	11	9	12	7
One More Time	13	9	14	17
Smooth	14	9	14	16
Crazy In Love	17	9	13	16
Help!	10	9	17	12
Respect	12	9	14	12
Back In the USSR	15	9	13	14
RMSE	N/A	4.98	3.08	2.95

TABLE 6.2: Table presenting an amount of segments in each song according to the ground truth and estimated by the checkerboard algorithm (by Foote, [36]), NMF based algorithm (by Kaiser, [37]) and ours.

repeates as often as verse and chorus, which is bound to introduce confusion to a person who hears the song for the first time and is asked to segment and label it.

In totall, we manually labelled 10 songs:

“The Number of The Beast” by Iron Maiden (Metal)

“Rock With You” by Michael Jackson (Soul/R&B)

“Blitzkrieg Bop” by The Ramones (Punk Rock)

“This Land is Your Land“ by Woody Guthrie (Folk)

“One More Time” by Daft Punk (Dance)

“Smooth” by Santana (featuring Rob Thomas) (Pop)

“Crazy in Love” by Beyonce (featuring Jay-Z) (Pop / Rap)

”Help!” by The Beatles (Rock)

”Respect” by Aretha Franklin (R&B)

”Back in the USSR” by The Beatles (Rock)

For each of the songs, we conducted 5 measurements to manually detect the boundaries. Once we have gathered the data, we averaged each bound to create ground truth with reference boundaries and labels.

To be able to fully evaluate the performance of our algorithm in comparison with state-of-the-art solutions already existing, we also gathered data produced by another two algorithms. The first one, described by Foote and Cooper [36], is a classic approach applying a ”“checkerboard” kernel over the diagonal of a self similarity-matrix (SSM). The other one is an approach published by Kaiser and Sikora [37], with use of non-negative matrix factorisation on only one feature.

Song	500ms	3s	deviation
The Number of The Beast	Foote	Foote	Ours
Rock With You	Ours	Ours	Ours
Blitzkrieg Bop	Kaiser	Ours	Ours
This Land Is Your Land	Ours	Foote	Foote
One More Time	Foote	Foote	Ours
Smooth	Foote	Foote	Ours
Crazy In Love	Ours	Ours	Ours
Help!	Ours	Foote	Ours
Respect	Foote	Ours	Ours
Back In the USSR	Ours/Kaiser	Ours/Kaiser	Ours

TABLE 6.3: Table showing the best performing algorithm in each category for all the songs.

We conducted two types of studies to determine the performance of our bound finding algorithm.

First and the most intuitive one is the comparison of amount of boundaries detected. If an algorithm returns a smaller amount of segments it means that most probably its detection system is not sensitive enough - it merges some of bounds into bigger ones. And vice versa - if the algorithm returns more boundaries than the ground truth, it means it picked on song segments that differ in a less obvious way and, hence, its output is too granular.

The results of our investigation can be seen in Table 6.2. By calculating the root mean square error for outputs of every algorithm, we noticed that our algorithm has a slightly better performance than the simple non-negative matrix factorisation one. On the other hand, checkerboard designed by Foote and Cooper consistently returned 9 labels, and hence yielding the worst performance.

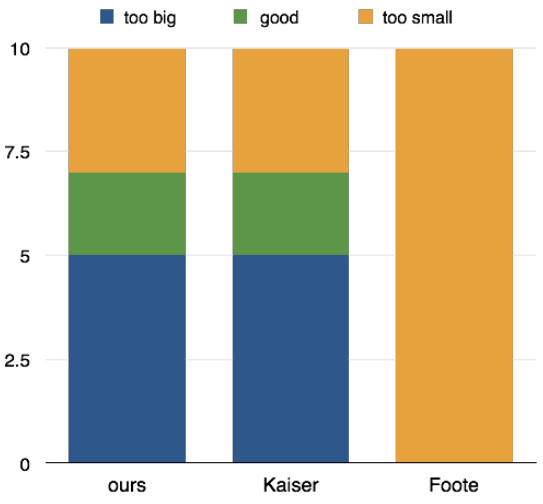


FIGURE 6.3: Figure presenting the amount of each type of errors produced by all the evaluated algorithms.

Apart from observing the RMSE of the boundary number, we reasoned about the distribution of the results. In particular, we believe that the situation when the algorithm is over-segmenting, returning more bounds than there are in the ground truth, is better than the one when the structure retrieval system is not sensitive enough and merges boundaries that do not belong together. This is especially true in case of our application, where if more boundaries are returned, the mood detection

becomes more accurate at expense of efficiency. On the other hand, with every lost bound we lose precision in the mood detection in two neighbouring segments.

If we look at Figure 6.3, we can see that in the light of our previous observations, the checkerboard yields a dramatically worse performance than nmf and our algorithm. In contrast, the latter two exhibit the same proportion of over- and under-segmenting.

Another measurement of success of our boundary retrieval is computation of boundary detection hit-rate. A hit is counted whenever a reference boundary is within a certain window of an estimated boundary. An important thing to note is that each boundary should be matched at most once, otherwise we can get some irrelevant results. For instance, if we are estimating the algorithm with window size 3 seconds, in the ground truth there are two boundaries 6 seconds apart and our algorithm detects only one right in the middle, if we do not exclude the boundary once compared with the first one in the ground truth, the performance of our algorithm will be ranked as much better than it actually was.

In addition to computing the hit-rate for windows of sizes 500ms and 3s, we also computed the median deviations between the reference and estimated boundaries of the song. In particular, we focused on median time from each reference boundary to the closest estimated boundary.

We implemented our evaluation benchmark using Python *mir_eval* library [54]. Again, to best evaluate performance of our algorithm, we compare the results it yields with results of the checkerboard and NFM based ones. Our findings can be seen in Table 6.3.

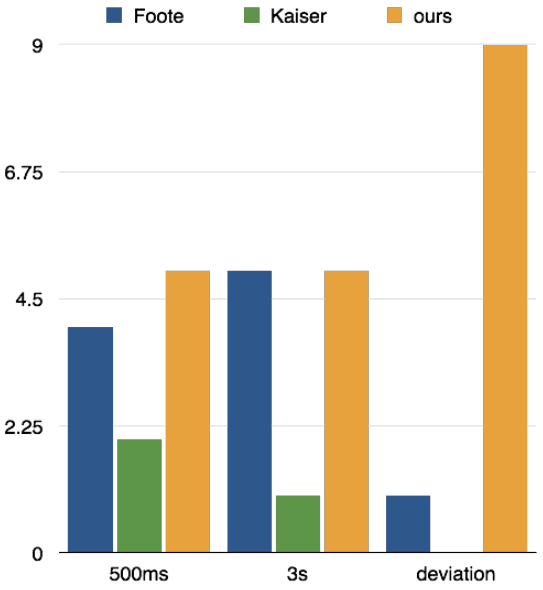


FIGURE 6.4: Chart presenting amount of times every algorithm excelled in each of the categories - hit-rate with window sizes of 500ms and 3s and the median deviation from the ground truth.

As we can see in the Figure 6.4, in the majority of cases, the performance of the algorithm designed by Kaiser falls behind the other two methods. It ranked as first twice when calculating the hit-rate with the window size w=500ms, once for the w=3s and it never excelled in the deviation category. On the other hand, the algorithm designed by Foote and Cooper yields similar overall performance as ours when evaluating hit-rate with w=3 and slightly worse for 500ms. This most probably is caused by the fact that in many cases our algorithm managed to find the segments but was a bit off, not qualifying for the 500s window. However, once the window got increased, its performance improved more than in case of the "checkerboard" algorithm. The exact results can be seen in section **??** in Appendix C.

In conclusion, we believe that the algorithm for boundary detection we created is suitable for our application and yields performance comparable or exceeding the state-of-the-art creations in its field.

6.3 Labelling

6.4 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.

Preliminary Research

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 18 people aged 17-25 asking about their past experience with music rhythm games. The questions and the results are presented in the Table 6.4. 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.

As we can see from the Table 6.4, the majority of young people surveyed did enjoy playing games to a similar extent. However, the results of the survey tell us that listening to music is almost unanimously beloved activity, with the highest average result and the smallest standard deviation. Majority of people play games quite often, but the lower average and standard deviation compared to the first question suggests that there are some people who enjoy playing games a lot but they do not spend that much time doing so, be it due to lack of time or other arrangements.

When it comes to rhythm-games specific questions, most people have played a game of such type before. A majority of people believed that having a set playlist was limiting their experience, however, there were some who did not mind this that much. However, when asked if the ability to upload their own music would improve their experience, everybody was either neutral of agreed - nobody was against the idea.

Question	Average	Stdev	Min	Max
Do you like playing games?	4.11	0.9	2	5
Do you like listening to music?	4.67	0.59	3	5
Do you often play games?	3.722	0.89	2	5
Have you ever played Guitar Hero or other rhythm music game?	3.88	1.84	1	5
Did you feel like the choice of songs was limiting you?	3.22	1	1	5
Were you able to load in your song of choice in there?	2.	1.41	1	5
Would you like to be able to load a song into it?	4.27	0.82	3	5
Did you feel like the game graphics were reflecting the emotions in the song?	2.78	1.06	1	4
Would you like the game to reflect the emotions in the song?	3.67	0.84	3	5
Was the game reflecting the section of the song you were in?	2.44	1.25	1	4
Do you feel it would be useful to know what section of the song is currently played?	3.5	0.86	2	5

TABLE 6.4: Table presenting the results of the preliminary questionnaire.

Majority, but no all surveyed believed that the rhythm game they played did not reflect the mood of the song they played, which can be deducted from the average below the neutral value 3 with the maximum value being four, so above the neutral. They believed it would be nice for the game to reflect the emotions in the song, but the need expressed was not as urgent as in case of the upload of their own songs. On the other hand, nobody opposed to such feature, which is reflected by the minimum value given being three.

Finally, we asked the surveyed whether they felt like the game was reflecting the built of the song, notifying them of where in the song they were. Most agreed that the games like that did not contain any sort of visualisation for the song segmentation. Majority of surveyed agreed that such feature would be useful, although a small amount of people believed it would not contribute in any way, answering with 2.

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.

Final Research

For our final research, we demonstrated our game to a group of 8 people aged 18-23 and asked them to fill in a questionnaire to describe their experience and thoughts on the game. 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. The questions and the results are presented in the Tables 6.5, .

Question	Average	Stdev	Min	Max
I knew what to do in the game straight away.	4.88	0.35	4	5
I needed hints to play the game.	2.25	1.58	1	5
I needed somebody to tell me how to play the game.	1.13	0.35	1	2
I liked the design of the menu.	3.86	0.99	2	5
I felt like the game interface was too crowded.	1.38	0.51	1	2
I felt like the interface of the menu was too empty.	1.75	0.89	1	3
I felt like i knew what each part was supposed to do.	4.75	0.46	4	5

TABLE 6.5: Table presenting the results of the final questionnaire concerning the game interface.

Question	Average	Stdev	Min	Max
The game was too difficult.	2.86	0.83	2	4
The game was too easy.	1.13	0.35	1	2
The buttons felt synchronised with the game.	3.75	0.46	3	4
I noticed the changes in the mood influenced the game interface.	3.25	1.04	1	4
I found the structure recognition useful.	4.00	0.35	3	5
I knew how to quit / pause / resume the game straight away.	4.88	0.83	4	5
I knew how I was scored straight away.	4.13	0.93	3	5

TABLE 6.6: Table presenting the results of the final questionnaire concerning the gameplay.

As we can see, a vast majority found the user interface fairly straightforward. The rest of the interviewed needed hints provided in the game to fully understand the game play and the flow of the use. Almost nobody felt like they needed someone to explain what they are supposed to do in the game.

More than half of the people liked the design of the menu - almost nobody felt like it was too crowded or too empty. In addition to this, the vast majority of the surveyed believed they could guess the purpose of the elements visible in the interface, with minimum response being 4.

The second set of questions was designed to find out about users' thoughts on the game itself. In the survey, we asked what the interviewees thought about the game's difficulty. The results can be seen in Table 6.6. Nobody believed the game was too easy, with an average answer of 1.13 and maximum one being 2. The majority of the surveyed also thought that the game was not too difficult, although the average response was much closer to neutral. We believe that this is a desirable outcome, as the game is supposed to pose a challenge to the player or they will quickly become bored with it.

Question	**Average**	**Stdev**	**Min**	**Max**
I think the application does not require previous computer experience to be used properly.	4.63	0.52	4	5
The buttons were designed in such way that the user can quickly become familiar with the game environment.	4.38	0.52	4	5
The screens are well structured and their design is clear, aesthetic and attractive.	4.00	0.76	3	5
The amount of on-screen texts is not excessive.	5.00	0.00	5	5
The fonts are clear and readable.	3.38	0.74	3	5
The combination of colours is pleasing.	4.38	0.74	3	5

TABLE 6.7: Table presenting the results of the final questionnaire concerning the overall experience when playing the game.

We also focused on the assessment of individual elements that create the game experience and make it stand out from all the publications available on the market. The incorporation of the predominant melody detection as well as the button generation algorithm were referred to in a question about synchronisation between the music and the notes that come up on the screen. Every person asked either agreed or was neutral in response to the question. We believe the variation in the answers is due to different choice of songs the surveyed decided to upload. If the song did not have a one definite predominant melody from a single source, the buttons would become less predictable. We are satisfied with this score as the fact that nobody gave it less than 3 implies that the algorithms fulfill their purpose and generate relevant outputs in general.

The next question regarded the mood detection. Although the average response suggests that the users did notice the changes in the visuals triggered by the mood changes in the music, in fact, many people felt indifferent about this feature. When asked what they thought the reason was, they stated they were too focused on trying to ace the buttons to pay attention to the animation in the background.

We also asked about users' opinion on the structure retrieval system and its incorporation. Most people found it useful, claiming it helped them track where in the song they were. We believe the feature was a success as the lowest score it received was a neutral one.

When asked about the controls, the all the users agreed that they were easy to find and use. In addition to this, they all felt that they new how the scoring worked and what to do to improve their results.

The final set of questions given to the surveyed asked them about their overall experience and thoughts on the application. Every person questioned believed to some extent that the application we created does not require the users to have previous computer experience to play - there was no answer below 4. This is really important, especially for a game, which we believe should be available and easy to use to everyone who wants to play it.

The surveyed also appreciated the design of the buttons claiming their design helped to understand the flow of use. They also believed that the structure of the screens was logical and visually pleasing. In addition to this, everybody unanimously agreed that the amount of the on-screen text was not excessive. This is encouraging, as very often games repel people by forcing them to read long lines of text. The colour scheme also received a favourable review, scoring an average of 4.38.

Most people believed that the fonts were clear and readable, but a few believed that a small increase in the font size would improve the user experience.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

Through this project we wanted to develop a game whose gameplay was influenced by the musics. By extracting We wanted do make sure that it would be intuitive and fun to use by all music lovers, regardless of their personal preferences in genres or bands. The successful completion resulted in 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.

The project also demonstrated that the state of the art in music analysis can be reliably and efficiently used in real world systems. The successful incorporation of the melody extraction and the design of the algorithm for mapping the it to a series of buttons on the screen showed that music analysis systems are not just the domain of research. In addition to this, thanks to applying smoothing algorithms we successfully managed to introduce variation in the difficulty of the playable songs generated.

In addition to this, we successfully designed an algorithm for boundary detection that uses both Mel-frequency cepstrum coefficients (MFCCs) and harmonic pitch class profiles (HPCPs) to determine the possible structure bounds in a song that performs comparably or better than existing solutions. Moreover, we proposed a novel method for labelling the extracted song elements in an easy-to-understand way, making use of the estimated main melody pitches.

Last, but not least, we developed a mood extraction system to dynamically generate surroundings in the game. By approaching the emotion value as a continuous problem, we successfully trained a neural network to predict the arousal and valence values of the emotion in the music track. In contrast to most of the existing publications, we also tracked the changes in the mood by retrieving its values on a per-segment basis.

In addition to this, the application had many additional features that improve the user experience and make the use flow more intuitive, such as extraction of song information

from ID3 tags, management of the levels with preservation of their score for or even bonus points for good performance in the game or ability to change the key assignment when playing the song.

Overall the project can be considered a great success, achieving all its goals, and contributing new and valuable work to the field of music analysis.

7.2 Future Work

Many interesting areas of further work exist for this project. One of the more intuitive ones would be to port our game to mobile devices. First we would focus on migration to iOS, as thanks to use of Swift it should be more straightforward than in case of other devices.

Morevoer, a thought could be put into implementation of the multiplayer levels. This would most probably require the users to use some other means of controlling the game rather than the keyboard, for example a gamepad. In addition to this, the game could be extended to make use of Internet connection in exchange of achievements and or created songs. This could be further used to enable multiplayer songs with people all around the world.

Bibliography

[1] I. Bogost. *How to Do Things with Vidoegames*, page 36. University of Minnesota Press, 2011.

[2] Guitar Hero sales article. URL http://arstechnica.com/gaming/2009/01/guitar-hero-iii-first-game-to-reach-1-billion-in-sales/. Accessed: 2015-06-01.

[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] Y. Yang, Y. Lin, Y. Su, and H. 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. URL http://en.wikipedia.org/wiki/Music_video_game. Accessed: 2015-06-01.

[6] Dance Dance Revolution screenshort. URL http://cdn-static.gamekult.com/gamekult-com/images/photos/00/00/34/99/ME0000349967_2.jpg. Accessed: 2015-06-01.

[7] Music video game definition. URL http://psp.about.com/od/pspglossary/a/Music-Game-Definition.html. Accessed: 2015-06-01.

[8] Internal Section screenshot. URL http://www.hardcoregaming101.net/internalsection/is2.jpg. Accessed: 2015-06-01.

[9] SimTunes screenshot. URL http://i.ytimg.com/vi/9CLYALO930k/hqdefault.jpg. Accessed: 2015-06-01.

[10] Rhythm game definition. URL http://en.wikipedia.org/wiki/Rhythm_game. Accessed: 2015-06-01.

[11] Guitar Hero screenshot. URL https://images-na.ssl-images-amazon.com/images/G/01/videogames/detail-page/gh3.lor.03.lg.jpg. Accessed: 2015-06-01.

[12] Guitar Hero controller photo. URL http://www.pixelplayer.se/bilder/texter/rec_8576_1_20081117.jpeg. Accessed: 2015-06-01.

[13] J. A. Moorer. *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer*. PhD thesis, Stanford University, 1975.

[14] 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. Accessed: 2015-06-01.

[15] W. M. Hartmann. *Signals, Sound, and Sensation*, pages 145, 284, 287. Springer, 1997.

[16] *Collins English Dictionary*. Collins, 2007. Ninth edition.

[17] 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.

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

[19] Short Time Fourier Transform. URL http://www.originlab.com/index.aspx?go=Products/Origin/DataAnalysis/SignalProcessing/STFT. Accessed: 2015-06-01.

[20] J. O. Smith III. *Spectral Audio Signal Processing*. W3K Publishing, 2011.

[21] J. C. Brown and M. S. Puckette. An efficient algorithm for the calculation of a constant Q transform. In *The Journal of the Acoustical Society of America*, volume 92, 1992.

[22] 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.

[23] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine*, 31(2):118– 134, 2014.

[24] Viterbi A. J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. Section IV.

[25] Vibrato definition. URL http://en.wikipedia.org/wiki/Vibrato. Accessed: 2015-06-01.

[26] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[27] aihorizon. URL http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.html. Accessed: 2015-06-01.

[28] D. Golmohammadi. *A Decision Making model for Evaluating Suppliers by Multilayer Feed Forward Neural Networks*. PhD thesis, West Virginia University, 2007.

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

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

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

[32] Y. Feng, Y. Zhuang, and Y. Pan. Music information retrieval by detecting mood via computational media aesthetics. In *Proceedings of the 2003 IEEE/WIC International Conference on Wev Intelligence*, pages 235– 241, Washington, DC, USA, 2003.

[33] W. Duckworth. *A Creative Approach to Music Fundamentals*, page 319. Schirmer, 2012.

[34] J. Covach. *Form in Rock Music: A Primer*, pages 71–71, 74–75. http://www.personal.kent.edu/ sbirch/Theory/213412005.

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

[36] J. Foote and M. Cooper. Media segmentation using self-similarity decomposition. In *Proc. of SPIE Storage and Retrieval for Multimedia Database*, number 5021, pages 167–175, San Jose, California, USA, 2003.

[37] F. Kaiser and T. Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *Proc. of the 11th International Society of Music Information Retrieval*, pages 429–434, Utrecht, The Netherlands, 2010.

[38] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. SIAM International Conference on Data Mining*, pages 606–610, 2005.

[39] O. Nieto and T. Jehan. Convex non-negative matrix factorization for automatic music structure identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 236–240, Vancouver, British Columbia, Canada, 2013.

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

[41] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 51–54, Vienna, Austria, 2007.

[42] N. N. Vempala and F. A. Russo. Predicting emotion from music audio features using neural networks. In *9th International Symposium on Computer Music Modelling and Retrieval*, London, 2012.

[43] 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.

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

[45] 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.

[46] Affective key characteristics. URL http://www.wmich.edu/mus-theo/courses/keys.html. Accessed: 2015-06-01.

[47] 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.

[48] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

[49] List of songs considered the best. URL https://docs.google.com/document/d/1AOSL6_j42EbaY6WPhSYUsfHgeYGDufRTkQ0a-4QhPA8/edit?usp=sharing.

[50] M. Müller. *Information Retrieval for Music and Motion*, page 65. Springer, 2007.

[51] Clustering Algorithms, CS345a: Data Mining, Stanford University, May 2015. URL http://web.stanford.edu/class/cs345a/slides/12-clustering.pdf.

[52] Swift documentation. URL https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0. Accessed: 2015-06-01.

[53] Sprite Kit documentation. URL https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html. Accessed: 2015-06-01.

[54] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Conference on Music Information Retrieval*, Taipei, Taiwan, 2014.