Pattern Extraction & Rhythmic Similarity of Electronic Music

Joseph Carver

January 16, 2025

Project Supervisor: Dr. Sasan Mahmoodi
Second Examiner: Dr. Richard Crowder

Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

A project report submitted for the award of BSc Computer Science

Abstract

The vast majority of modern electronic music played by DJs today can be categorised into one of several distinct sub-genres, or styles, for example; House, Electro, Techno or Jungle. Each of these styles can be almost scientifically defined by its rhythmic profile, which includes, amongst other characteristics tempo, meter, and a range of more expressive features. A large majority of work into audio classification and identification focuses on analysing tonal and textural elements, with rhythmic representations merely an afterthought, but effective automated rhythm fingerprinting and matching software could easily find application in a DJs digital toolbox. Most previous work that has taken a rhythm-oriented approach to music information retrieval (MIR) has focussed on classifying standard and Latin ballroom dance works, with some areas of interest only recently having been studied in the context of 'EDM', a far less rhythmically diverse genre.

This project explores the performance of a content-based similarity system that makes use of rhythmic feature descriptors to provide suggestions of perceptually similar pieces of music to a query piece. The implementation utilises a simple metrical profile to locate salient patterns in a piece of music and extracts the patterns to form a type of audio fingerprinting. Evaluation of the investigation on an in-house dataset found that these patterns can be compared efficiently with one another to give meaningful rhythmic suggestions from a library of electronic music of varying styles, although some user correction is required for the system to be fully effective. A simple interface is presented to facilitate these corrections.

Contents

1 Introduction

Although we do not often realise it, whenever we hear a piece of music we perform some degree of classification on the audio, based on features present in the recording or performance. These features generally fall into one of two categories - tonal or rhythmic. For any piece of audio represented digitally, signal processing techniques enable automatic extraction of a wide range of features belonging to both of these categories. By applying distance measures to dominant features in individual works, it is possible to group tracks that have similar characteristics, and therefore, in theory sound similar.

The process described above has clear utility in a content-based analysis and recommender system. With such a broad range of features that can possibly be extracted, the key to a meaningful analysis procedure lies in determining which features contribute most to the auditory profile of a track, i.e. the way in which it is perceived by the human ear, and how these perceptions contribute to defining the genre boundaries we recognise. In the domain considered here - modern electronic dance music - rhythmic characteristics such as tempo and outer/inner-meter are considered by many to be sufficiently indicative of sub-genre that they can be used to distinguish between most cases. The aim is therefore to extract and analyse rhythmic features from songs and use the vectors obtained to classify and rank collections of music.

It is important to remember that there will inevitably be some contention amongst listeners as to which specific sub-genre certain tracks belong to. This project is not aiming to resolve such disputes, but rather will provide a means to discover rhythmically similar tracks in an existing collection. With that in mind, the primary goals of this project are as follows:

1. To investigate the extent to which extraction and analysis of rhythmic feature indicators can successfully classify a range of modern electronic dance music pieces into their relatively broad sub-genres (Electro, Footwork, House, Jungle, Techno, UK Garage)

2. To build a scalable tool that will recommend rhythmically similar tracks from an existing library

The remainder of this project report proceeds as follows: Chapter 2 discusses the background reading that contributed to important decisions throughout the project. Chapter

3 outlines the approach taken to software development and the important considerations in this aspect. Chapter 4 describes in full the implementation, that is, the specific techniques that were used to solve many of the problems introduced in ch. 2. Chapter 5 sets out the strategy used for evaluation before presenting the results obtained, and chapter 6 discusses in detail the implications of the results. Finally, Chapter 7 reflects on the findings and offers some areas for future development.

2 Background Research

2.1 Rhythm Perception

In order to design a system that can accurately describe music by rhythm, it is first necessary to understand the way in which the human ear perceives rhythm. Honing discusses in (11) how the perception of a performed rhythm generally derives both an overarching rhythmic category (tempo, meter) as well as "expressive timing", which refers to more stylistic characteristics. Experiments carried out in 1973 (9) explored the multidimensional nature of rhythmic characteristics, concluding that, amongst others, "rapidity", "uniformity-variation" and "forward movement" are all defining factors in a piece of music's rhythmic profile. Even earlier than this, Cooper & Meyer (3) wrote that "to experience rhythm is to group separate sounds into structured patterns", and in doing so succinctly summarised one of the trickiest problems facing many music analysis systems.

2.2 Automated Feature Extraction

Audio feature extraction in general has been explored in several contexts, most famously by Shazam in their largely-successful efforts to automatically identify tracks based on short, user-recorded snippets of audio (23). Genre-based classification by a variety of features has been studied in (22) and (13) with good degrees of success, although in both cases the dataset analysed included music from a much wider set of genres. In the latter, McKinney & Breebart found through thorough testing that "a feature set based on a model of auditory perception out-performs other current standard feature sets", referring in this case to auditory filter bank models. They also noted that "temporal variations of features are important for classification" regardless of the feature set used.

2.3 Features for Rhythm Analysis

Automatic genre classification by rhythmic analysis alone has not been studied in as much depth. In 2001, Foote & Uchihashi introduced a novel approach of computationally representing rhythmic features in music based on self-similarity, which they called the beat spectrum (8). The algorithms developed here were able to "reveal both the tempo and the relative strength of particular beats". Dixon et. al expanded on this approach in (6) by identifying the most prominent bar-length rhythmic patterns in each piece, and using these patterns as the bases for inter-track distance measures. In this study, the rhythmic profile of a pattern was represented by "the amplitude envelope of the signal between the

start and end points of the bar", and was analysed in conjunction with tempo and other derived features to achieve a 96% classification success rate. Additionally, they noted that extracting patterns from specific frequency bands could potentially yield improved results; this idea is explored in this project.

The work of Chew et. al a year later (2) had the same goal of distinguishing between pieces with the same tempo and outer-meter, but took an alternative approach to examining inner-meter. By assigning each note with a metric and spectral weight (14) corresponding to its significance in a local meter, the system was able to classify Standard and Latin ballroom works with around 70% accuracy.

There have been some recent developments in the use of features extracted from recognised rhythmic patterns. In (20), Smith introduced a new model of rhythmic profile entitled the Metrical Profile, which represents high-level features such as syncopation and is computed over a pre-determined prominent pattern. The model is defined with the intention of closer modelling human perception of a track whilst maintaining its viability as the basis for similarity measures. Their work was able to classify the Standard and Latin Ballroom dataset with 67% accuracy, although the feature vector was very high in dimensionality.

In terms of technologies available, work published in 2007 presents a set of entirely modular functions written in Matlab and designed purely for the extraction of music features from audio (12). Lartillot and Toiviainens MIRToolbox can extract with startlingly-simple syntax a comprehensive range of musical features, including pulsation and periodicity, making the toolbox ideal for this study.

2.4 Tempo Estimation

Settling on an appropriate and accurate tempo estimation algorithm proved much more time-consuming than expected. Exact tempo estimation is far from a solved problem, and a thorough analysis of approaches in (26) shows that the best-performing algorithm can achieve 91% accuracy, with most achieving somewhere between 50 and 70%. The `MIRToolbox` implementation, `mirtempo`, was found to produce 75% accurate results, roughly consistent with the findings in this project.

The study in (26) concludes that due to the nature of the most common errors (double error and 4/3 error), heuristic approaches to combining methods can achieve significantly higher accuracy.

2.5 Beat Tracking & Segmentation

The problem of accurately tracking beats and recognising phases in a piece of music is central to the study of content-based analysis. The performance of many integral algorithms and procedures is directly affected by the accuracy with which the system has segmented a track, a process which relies on correct knowledge of bar phases (start-times) and periods (durations).

In particular, as rhythmic feature analysis tends to focus on short repeating patterns (18), the desire to recognise and extract salient patterns comes to the fore. If the extracted pattern does not correspond to the pattern as a human perceives it, then any feature computed from the pattern will not be descriptive of the track in a wider context, regardless of its prominence in the track. To clarify this, consider the diagram below consisting of two identical rhythmic patterns that have been delimited differently.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
low	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
mid	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-
high	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
low	-	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-
mid	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-
high	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1

Figure 1: *Columns correspond to measures in a bar. A value of 1 in a position indicates a significant rhythmic event at that point.*

The two rhythms displayed in figure 1 can be recognised as identical to each other except for a rightwards shift of 1/16th note, or 1 semiquaver. This kind of error can result from imperfect segmentation, and can present a significant issue for analysis algorithms, as

these rhythms would be treated as dissimilar.

Although an algorithm can usually determine to an acceptable degree of accuracy the location of beats in a piece, it is far more complicated to identify algorithmically which of these is a downbeat and marks the start of a bar - a task which is perceptually straightforward for most humans.

In a 2013 analysis of 'EDM' music, Rocha et. al (19) combined initial downbeat detection with timbre-based novelty measures and some musically informed rules to achieve up to 51% segmentation accuracy (within 0.5s) on a purpose-built EDM dataset. Downbeat detection in this case consisted of first applying a band-pass filter to isolate frequencies commonly containing the kick drum, and then searching for peaks in an onset detection curve centered around the first point of high response of an RMS computation.

Panteli had some success in (16) by using the segmentation method introduced in (19) to divide a signal into meaningful regions, before applying a downbeat detection algorithm based somewhat on musically informed rules. The algorithm achieved 51% accuracy at locating the downbeats within 50ms - suggesting about half of the tracks could be susceptible to incorrect bar phase information. In their case, the issue was mitigated as the system calculated similarity based on a number of features, crucially including a periodicity measure computed over an arbitrary length section of the track.

In (6), beat detection as a prerequisite to segmentation was achieved with the help of BeatRoot (5; 4), a system that algorithmically estimates beat positions and periodicities in a track and is supported by an interface to allow for user-corrected values. Their beat-tracking algorithm works by measuring frequency of occurrence of durations between various note event onsets and hypothetically comparing sequences of these onset events to an existing tempo estimation. Combinations of note events that are predictable and reoccur according to some multiple of the tempo estimation are grouped together to form a rhythmic structure, from which the beats can be derived.

To summarise research into this issue, even a system that is able to perfectly label the location of beats in a track is unlikely to be capable of correctly identifying the metrical boundaries (bars) that encapsulate the prominent rhythm. Any system reliant on wholly accurate representative rhythmic patterns must therefore include some way for a user to

correct the system's estimations.

2.6 Measuring Similarity

The final aspect to consider then is how to use the detected features to identify rhythmically similar songs. The field of content-based music retrieval offers a good amount of research on which to base an approach to this problem. (24) presents a fairly rudimentary approach to querying a database through similarity features, but falls short of analysing in any real depth the rhythmic characteristics of tracks.

Pampalk et. al take a more graphical approach in (15) by using Self-Organising Maps to cluster tracks into "Islands of Music" that theoretically resemble genres. The approach taken here relies more heavily on a detailed examination of rhythmic features, and in fact contributed to some of some of the algorithms implemented in the MIRToolbox. The weather maps outputted by this system provide a convenient way in which to evaluate the performance of the classifier that is not reliant on specific sub-genre labels that can cause contention.

An in-depth comparison of alternative rhythmic similarity measures carried out by Toussaint (21) made use of phylogenetic trees to evaluate the efficacy of common distance measures at rhythmic categorisation in a variety of vector spaces. The intuition behind this approach is that a good similarity measure will provide an insight into "the structural inter-relationships that exist within families of rhythms", which contrasts to the usual technique of evaluation with reference to human perception. Each phylogenetic tree generated by a distance measure for a family of rhythms was examined for characteristics such as goodness of fit, clustering, and any ancestral structure, with chronotonic and edit distance emerging as the best overall rhythmic similarity measures.

Toussaint also found that the Hamming distance performed very well when classifying binary rhythms. The Hamming distance has its roots in information theory and error checking, but can be computed between any two equal-length binary vectors to denote the difference between them, i.e. the percentage of elements of vector A that are not the same in vector B.

The Hamming distance as described above is naturally suited to rhythm comparison, but

has the drawback of not measuring the magnitude of any mismatches between vectors, as it considers each vector element entirely separately from one another. The Levenshtein, or edit distance is a variation on Hamming designed to ameliorate this flaw, by measuring distance as the amount of operations required to convert vector A to vector B, where the operations permitted also include left and right shifts of elements in each vector.

3 Approach

This section describes the most significant decisions made throughout the development process and how they have been influenced by the goals and requirements of the project. Also presented is an overview of the design of key system components.

The original intent of this project defined two distinct goals - to explore the capabilities of algorithmically generated rhythmic feature descriptors, and to implement the developed feature within a content-based suggestions system supported by an intuitive interface. With that in mind, the following requirements are identified.

Functional Requirements

1. A user must be able to select tracks from their file system for analysis
2. The system must support individual and batch analysis of files
3. The interface must provide straightforward options for a user to manually correct any analysis files
4. The interface must offer some visualisation of analysis files to the user
5. The user must be able to specify a track and view suggestions of similar tracks
6. The system should allow for specification of weights for the three frequency bands when computing similarities

Non-Functional Requirements

1. The system should be scalable up to a library of several hundred tracks
2. The system should store all analysis files on disk to facilitate an interactive and responsive interface
3. Analysis files stored by the system should be very small compared to the the original audio files

3.1 System Architecture

The main challenges associated with implementing the content-based analysis system as an interactive and intuitive piece of software mainly concern managing an optimal execution flow throughout analysis. When dealing with audio data, generally large in size compared to text or image data, it is vital to consider the time and space requirements that come with processing and storing analysis files. With this in mind, some key design decisions are outlined below.

3.1.1 Track Data

All data and files associated with any given track are stored in a `TrackData` object. Class instances are created with a path to the audio file to be analysed, from which the constructor generates a track name, used for identification purposes, and creates an info directory if it doesn't already exist. If a file with this name has been analysed before, the constructor finds the analysis files and loads their data from disk. Each analysis file is by nature several magnitudes smaller in size than the original audio file, allowing the system to hold all files in memory during use. The properties of a `TrackData` object can be categorised based on their purpose:

Basic Properties: `TrackName, OriginalPath, PathToInfoDir`

Feature Properties: `Tempo, TempoCands, BestBar, BestBarLoc, Amplitude, AttackLength_s`

Temporary Properties: `SimilarTracks, TrackWaveform`

Dependant Properties: `TempoExists, BestBarExists, AmplitudeExists`

The temporary properties are described as such because they are computed on demand at run-time, and are not saved to disk storage. Lists of similar tracks are generated dynamically and displayed immediately, depending on user-specified parameters, so there is no need to store these. The waveform object is not stored due to its size and is created only when necessary, as depicted in section 3.1.2.

The class facilitates updating and accessing of analysis files to and from persistent storage through its methods `updateDiskData` and `getFromDisk`, as well as simple methods to check if any given analysis file exists. This functionality increases flexibility of usage of the class outside of the global context, for example providing a framework for carrying out efficient evaluation through script usage. All `TrackData` objects are held in a global

`TrackArray` instance, which is created at start-up time from a persistently stored list of file-paths and is accessed by the GUI-governing class.

3.1.2 Flow of Execution

Amongst other features, the interface - illustrated in section 3.2 - provides a straightforward means for a user to select files for analysis, perform individual or batch analysis, correct any errors in the data generated during analysis, and of course display similar track suggestions. The analysis processes are contained in functions separate to the GUI class which are called hierarchically starting from the callback function for the corresponding button. Flow of execution for a full analysis of a track is shown in figure 2.

A handful of global variables are maintained throughout execution to ensure all standalone functions have access to any required information. These include an `ActiveTrack` object that determines which track's data is passed to the analysis functions and displayed on the interface, as well as absolute paths to important disk locations, for example that of the file list . There are also a number of files containing helper functions with re-usable or bulky code that have been separated from the GUI class.

One of the most computationally expensive steps in this process is computing the waveform object, which is achieved using `miraudio`. The full waveform is required for both tempo estimation and bar extraction, so for efficiency purposes we do not regenerate the waveform before each of these. This is facilitated by a `TrackWaveform` property of the `TrackData` class, which is instantiated whenever features need to be computed, and destroyed (set to empty) immediately after. Computing the amplitude envelope from which the eventual feature is extracted does not require the entire track waveform, only a waveform for the "best bar" generated during pattern extraction.

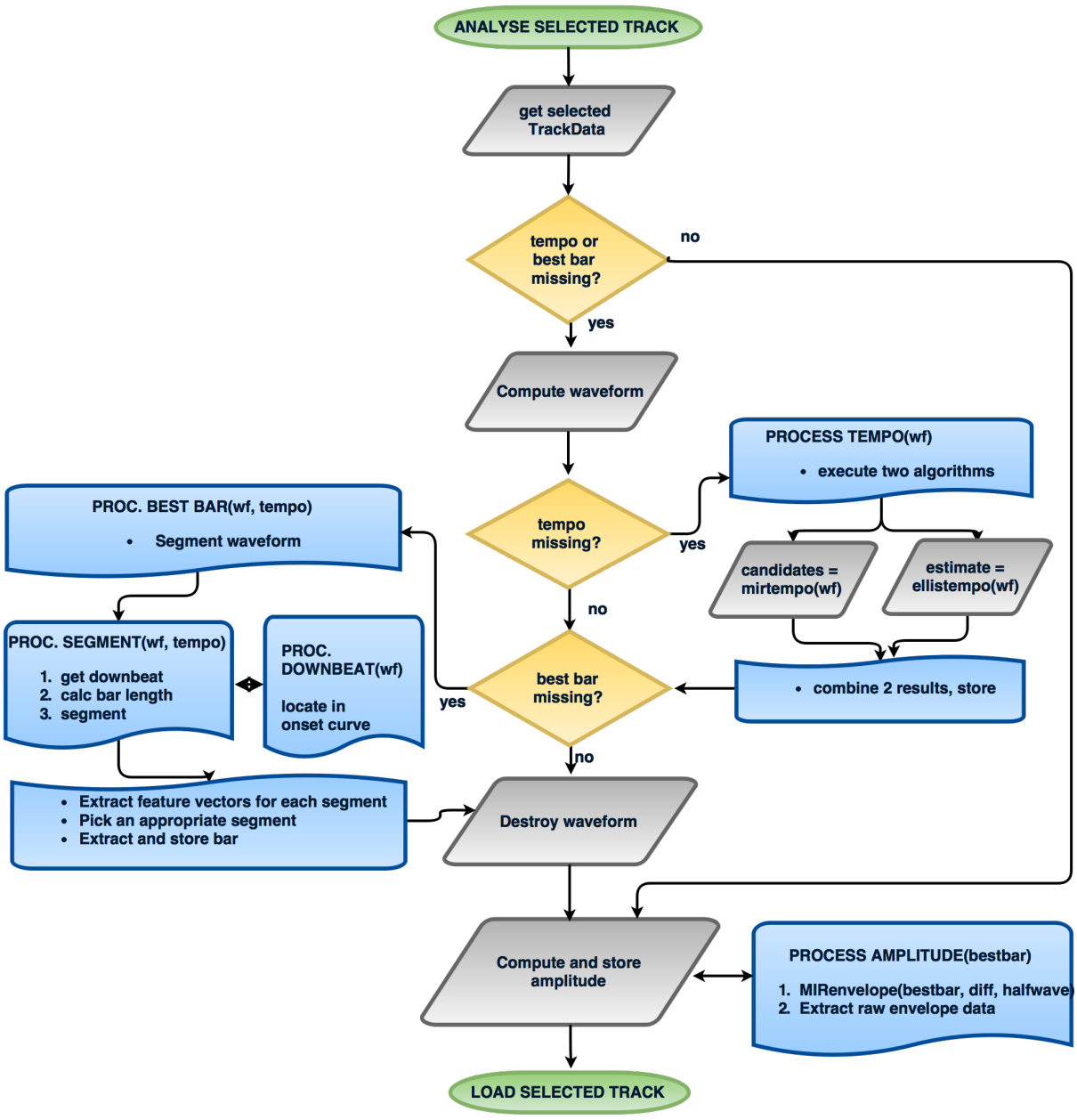


Figure 2: *Flow chart depicting execution flow required to fully analyse a track*

Another major advantage of this approach to execution flow is granted by separation of the complex audio processing operations from the main control flow. This allows for straightforward interchanging of different analysis algorithms, requiring very few alterations to existing code.

3.2 Interface

Developed alongside the underlying feature extraction and analysis framework, the graphical user interface not only facilitates the process but can drastically improve the results produced. A number of controls on the interface allow a user to correct crucial parameters and data that the automated system has miscalculated.

Matlab's GUIDE framework houses the interface, meaning the front and the back-end components run in the same environment, which is hugely advantageous for a project like this.

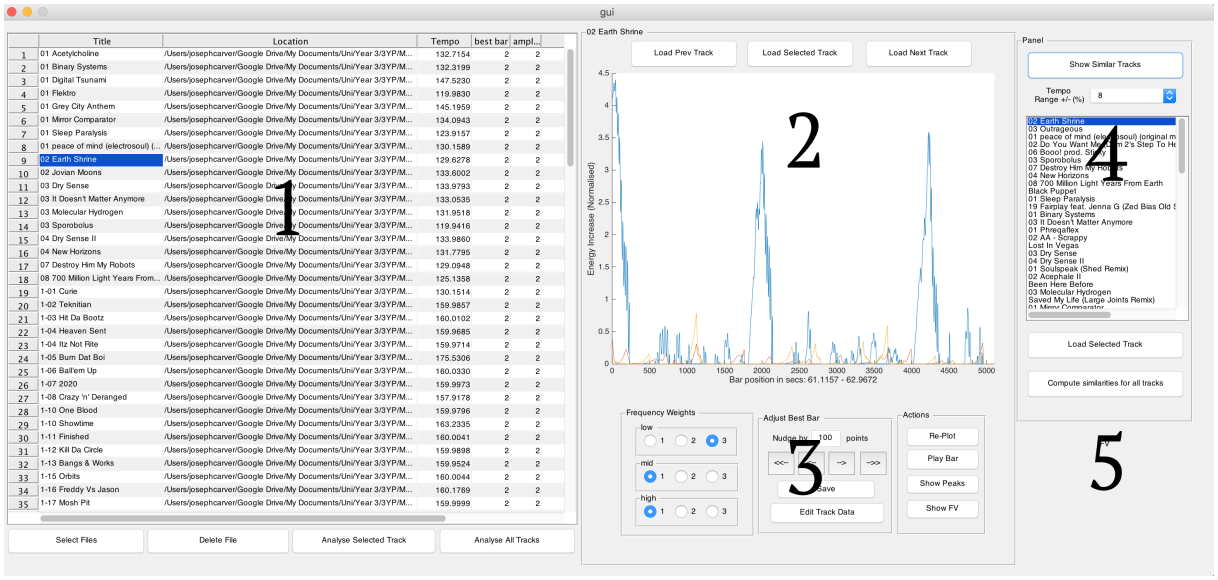


Figure 3: *Graphical User Interface for the system.* ***1*** *is a list of files tracked by the system.* ***2*** *displays track data for the selected track,* ***3*** *allows a user to edit it.* ***4*** *displays a list of similar tracks.* ***5*** *displays a preview of the feature vector, mainly included for evaluation purposes*

Side-by-side development of the front and back-end components enabled an incremental approach to designing and refining the interface features and layout. Design decisions were influenced directly by commonly required operations in a typical work-flow, and as a result the interface design closely reflected a verified use case. This use case is described below, with reference to the labelled components:

- User clicks 'Select Files' and browses their file system for tracks to analyse. Information regarding chosen files is displayed in **1**, displaying track name, path and any analysis files already generated
- User selects a file in **1** and clicks "Analyse Selected Track". The back-end scripts compute any missing analyses for the track
- User clicks "Load Selected Track" and **2** updates to show the bar-length amplitude envelope for the track
- Track information and parameters can be adjusted using the controls in **3**. Includes visual and aural feedback - user can play the audio clip and visualise the corresponding feature vector (displayed in **5**)
- Once satisfied with the positioning and length of the representative bar, the user can generate similarities to be displayed in **4**. The weighting applied to each frequency band is determined by the selections in the left-most button group of **3**

4 Implementation

This section describes in detail the resultant software, and the technologies and techniques used to implement the required feature extraction operations. First we present a short summary followed by a description of the key analysis process, corresponding to the blue boxes in 3.1.2. Next we discuss specificities of the similarity procedure, before concluding with a closer look at the user interface.

4.1 Summary

The entire application is written in Matlab, and as initially planned, MIRToolbox forms the foundations of the system. Several important processes begin by extracting MIRToolbox objects from music files, but in all cases other signal and feature processing is required. The `mir` objects were found to be unnecessarily cumbersome in certain circumstances, as well as occasionally restrictive. As a result, it was regularly necessary to extract the raw data from these objects for more nuanced analysis and for persistent storage, whilst keeping in mind performance requirements.

The development process differed slightly from the planned process in that the front-end interface was produced alongside the underlying audio processing architecture. This change was made to facilitate incremental testing and tweaking to the system parameters, whilst ensuring a fully-functional system was never more than a few lines of code away.

4.2 Tempo

The approach taken to tempo estimation in this project reflects conclusions made in (26) that suggest the possibility of improvements in accuracy through application of a heuristic correction function. The `process_tempo` function in this implementation takes an instance of `TrackData` (3.1.1) and computes 2 separate tempo estimations before using the result of one to select the most appropriate candidate from the result of the other.

The first procedure is `mirtempo` which is computationally expensive but produces very precise results. The default configuration for this algorithm uses a well-established approach of measuring autocorrelation of the onset detection curve to detect periodicities, and produces 3 candidate tempos. Analysis of its output (section 5.2) reveals that ambiguities in rhythmic interpretation cause a number of tempo estimations to be erroneous,

as the algorithm chooses incorrect periodic groupings of onsets as being definitive.

The second algorithm is that introduced by Ellis in (7) which utilises dynamic programming techniques to compare sets of hypothetical beat times to an estimated global tempo. This implementation is far more efficient than `mirtempo` and produces two output candidate tempos that are accurate in a different way to `mirtempo`. The analysis of this algorithm, presented in section 5.2, corroborates the creator's own evaluation in demonstrating poor results when tested with a low acceptable margin of error, but good performance when the accepted tempo range is increased. This is to say that the algorithm generates tempo estimations that are usually within the correct region, but are rarely precisely accurate.

The final tempo estimation for a track is selected by comparing each pair of candidates from `mirtempo` and Ellis'. A track's tempo is chosen as the candidate from `mirtempo` that has the minimum distance to either one of the Ellis suggestions. The two tempo candidates produced by Ellis are in almost all cases at a ratio of 1:2 to each other (e.g. 65 and 130BPM), so the correction process should theoretically find a match regardless of the scaling decided upon by `mirtempo`. To ensure consistency across tracks, any tempo below 100BPM is doubled.

The analysis steps that follow tempo estimation require very precise estimations to generate truly meaningful patterns and features. The results from Ellis' algorithm are generally too imprecise and the results from `mirtempo` are too often wholly inaccurate to be used for these purposes. The "educated guess" principle applied here helps to correct many of those issues algorithmically.

4.3 Pattern Extraction & Feature Descriptor

A feature vector based on bar-length amplitude envelope windows forms the core of the investigation, and is used centrally in both the pattern extraction process and the content-based retrieval function.

The entry point for the pattern extraction procedure is the function `process_bestbar`, which takes a TrackData object as input. This function then calls subroutines for downbeat detection, segmentation, and feature extraction, all of which require only raw numeric

audio data.

The inspiration for this method of extracting rhythms is partly derived from (6), which used an amplitude envelope extracted from a bar-length pattern to represent each piece. The exact approach taken in this project however, differs slightly at each stage and is described below.

4.3.1 Segmentation

Before individual bars can be analysed and grouped, it it necessary to determine the start and end points of each bar. Accurate performance of the segmentation process relies entirely on the accuracy of 2 pieces of information about a track: its tempo (section 4.2) and the location in time of the first downbeat.

The method of computing downbeat locations is based largely on that introduced in (19). The audio input is analysed chronologically in 5-second windows, at each time applying a low-pass filter to the signal before computing the global-energy (RMS) in that low-frequency window. If the RMS response value exceeds a certain threshold, the window is deemed to contain a kick-drum. An onset detection curve is then computed for the window, and the location of the first kick-drum is determined as being the first point with an onset value above a threshold. The onset curve is also utilised at this stage to compute a value equal to the length of the kick-drum attack phase, which is stored persistently. Storing this variable for each track is vital to ensure that the positions of low frequency note events are correctly interpreted by the feature vector creator function.

Once an appropriate downbeat has been chosen, the locations in time of all bars in a piece are estimated by iteratively adding a bar-length - computed from the track's tempo - to the start location of each previous bar, beginning of course at the downbeat. A numerical array of bar start-times can then be used as input to the `mirsegment` algorithm.

In addition to the pre-computed track information, this approach relies on two main assumptions -

1. That the first instance of strong low-frequency energy in a track corresponds to the perceived first beat the bar.

2. That the quantised nature of electronically-produced music means each bar is exactly evenly-spaced and of equal length.

A more sophisticated approach, for example that in (6) of searching algorithmically for the exact location of each bar-start time within an expected region, could be explored, but would be unlikely to yield any improvement in performance for the dataset in question.

Similar to Dixon's approach, the bar-finding process relies on some user-correction to produce meaningful results. Once the algorithm's estimations have been made, a user can correct the bar-start locations with small adjustments made in either direction, or by entering an entirely new start time. Further information on these options is presented in section 4.5.

4.3.2 Pattern Extraction

Once the waveform has been segmented, the differentiated, half-wave rectified amplitude envelope is computed over three auditory channels for each segment, using a correctly configured `mirenvelope` function. This envelope gives precise local descriptions of increases of energy produced in each channel, and is therefore ideal for indicating the presence of percussive elements.

Application of a simple musically-informed rule concerning the typical structure of a prominent rhythm allows us to immediately discard bars containing patterns that do not have a kick-drum on their first beat. This step is implemented by examining a feature vector (section 4.3.3) produced for the low-band of each bar, and has the desirable side effect of also removing some bars that were not correctly delimited by the segmentation algorithm.

Once we have a set of relevant bars, a voting-style procedure is applied to generate a representative rhythm for a track. Each relevant bar contributes a vector for each of the three frequency bands, and the mean vector is computed for each band. A thresholding operation sets to 1 all the points in the mean vector that had been 1 in an appropriate proportion of the original relevant bar vectors, 0 otherwise.

In some contexts, it may be desirable to simply use the representative rhythm vector as a descriptor for the track. However, the intended application in this project requires that

we carry out some form of audio fingerprinting for each track. This entails a final step of comparing each relevant bar to the representative rhythm in order to identify any one bar in the piece that contains the desired rhythmic pattern. A weighted Edit distance is computed between corresponding frequency bands for each segment and the representative rhythm, where the weights correspond to the importance of patterns in that band to the rhythmic profile. For dance music, most of the rhythmic variations that define the styles we recognise are found in the low frequency components, so the weightings applied to the low, mid and high bands are 3, 1 and 1 respectively.

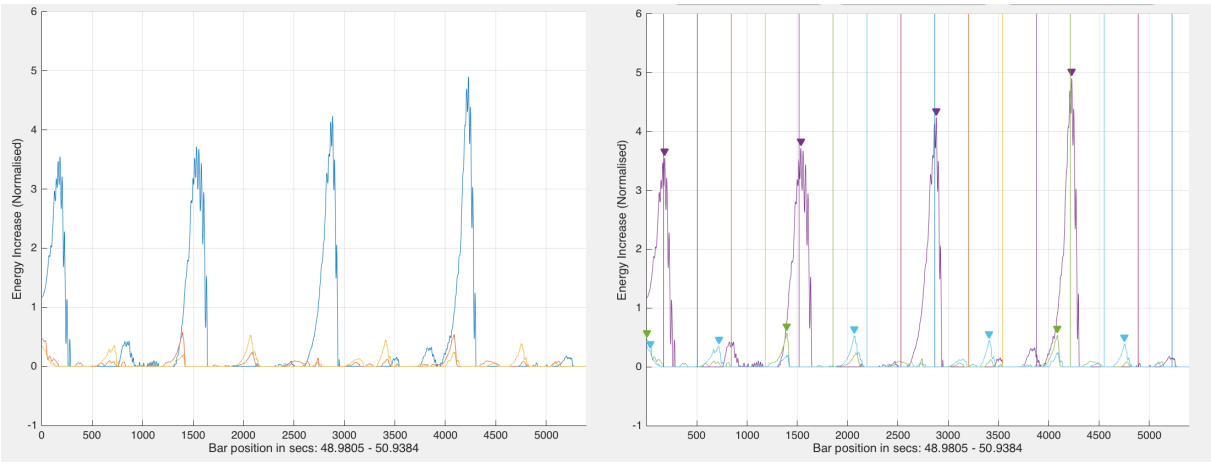
The start and end times of the pattern that scores the lowest distance from the representative vector are chosen as the boundaries for the track's representative, or "best" bar. This bar is extracted from the original audio waveform and transformed into the signal domain using `mirenvelope`. The resulting amplitude envelope is differentiated and half-wave rectified before the raw numeric data is extracted from the `mir` object and saved to disk storage, along with the roughly 2 second audio snippet containing the bar data itself.

4.3.3 Vector Representation

Feature descriptors are computed on a bar-by-bar basis from the differentiated, half-wave rectified amplitude envelope of that bar. This is intended to capture the notion of "metrical profile" introduced in (20) and expanded upon in (16). It is ideal for these purposes as it is small in size and very quick to compute given a bar-length waveform, but is capable of being highly descriptive in terms of rhythmic profile and can be easily compared algorithmically to other vectors using common similarity measures.

Vectors are extracted directly from observations on the differentiated amplitude envelope. First, we identify the locations at which the prominent peaks occur in the envelope for each of the three auditory channels. A linearly spaced vector with 16 points (1 for each semiquaver) is generated between 0 and the length of the bar. The actual peak locations for each channel can then be translated into relative locations and stored in a 16-element vector. Each element in the vector is a binary value, simply representing whether or not there is a significant increase of energy in the channel at that point in the bar.

An example below shows the 3-channel amplitude envelope for a track and its corresponding vector representation.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
low	1	-	-	-	1	-	-	-	1	-	-	-	1	-	-	-
mid	1	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-
high	1	-	1	-	-	-	1	-	-	-	1	-	-	-	1	-

Figure 4: *"Clap Ya Hands" - differentiated envelope, with and without peak indication for three frequency bands. The vertical lines on the peak display graph indicate the centre points of the 16 bar regions. Below, the generated binary feature vector corresponding to peak placement.*

4.3.4 Frequency Bands

The decision to divide each feature into 3 distinct frequency bands is guided by domain-specific knowledge. Prominent percussive elements in electronic music tend to fall distinctly into one of the three bands recognised here, where the low band contains kick drums, mid contains snares drums, or claps, and the high band comprises hi-hats.

Filterbank decomposition is achieved through use of `mirfilterbank`, which utilises a well-established simulation of the basilar membrane introduced in (17). This representation is found to be satisfactory in separating the three categories of percussive element, although there is undoubtedly some room for improvement (see section 7).

4.4 Similarity Function

Choosing an appropriate similarity measure for any application depends largely on the nature of the representation chosen to summarise each data point. In this case, each

track is represented by three binary strings of length 16. This feature demonstrates a massive reduction in space required to store necessary track data, and also ensures exact consistency of feature size across all tracks in the dataset. These properties of the feature help facilitate straightforward comparison through application of simple binary distance measures, such as Hamming, or Edit/Swap distance. The slightly more intelligent Edit distance offered a small increase in classification performance over Hamming in all cases, at an insignificant extra computational cost, and so is the distance measure chosen as the basis for all vector similarity operations.

Similarity between two vectors is considered for each pair of corresponding frequency bands, i.e. we compute low-low, mid-mid and high-high similarity. The value from each of these calculations is weighted according to the user-specified parameters from the interface component, before the three are summed together and returned as the distance between the two vectors.

When computing track-track similarities, a user specified tempo range is introduced. This check occurs before extraction of feature vectors and simply ignores any tracks that are not within the percentage range. The permitted tempo range is by default +/- 8%, chosen to reflect the range most commonly available to a DJ, and is inherently the range most suited to enhancing recommendation quality (see 5.4). The other options are +/- 16, 25, 50 and 100%(no restriction), corresponding to tempo ranges available on more sophisticated equipment.

4.5 Interface

Section 3.2 introduces the interface and the main use case that guided much of its development. In this section, some of the most significant actions and the operations they trigger are described in further detail.

A Matlab GUI file is the entry point for the application as a whole, On startup, the initialisation function loads a number of important pieces of information from disk and into memory. For example, the helper function `loadFileList` reads a `'\n'`-delimted text file and parses each file path to create a TrackData object, whicb is subsequently stored in the TrackArray.

The interface source file also contains functions that retrieve `TrackData` objects from the global `TrackArray`, update the relevant visual components with the requested data, and call routines to carry out analysis on tracks.

Select Files opens a file browser for the user to select files to be included in analysis, parsing the selected file paths and adding them to the stored `filelist.txt`.

Analyse Selected / All Files triggers the analysis process outlined in section 3.1.2 for 1 or all files

Delete Track searches the filelist for the track name of the highlighted track, and removes the entry. System needs restart for changes to take effect

Load Selected Track checks which row is highlighted in the track table and sets the corresponding `TrackData` as the `ActiveTrack`. The track's amplitude data for the representative bar is accessed and plotted, resizing the axes to match the length of the bar. Users can also load the previous or next track with corresponding buttons.

Tempo can be corrected in one of two ways. Selecting the `Tempo` column in the track table for a track displays a pop-up from which the user can select any one of the candidates produced by `mirtempo`. Alternatively, the user can enter a custom tempo by loading the track then using the "Edit Track Data" button. Each of these re-triggers the analysis process for that track using the new tempo.

Best Bar adjustments can be made incrementally with arrow nudge buttons. A user can adjust the bar position by 1 bar at a time, or by a small, user specified value. This value corresponds to the amount of points in the amplitude envelope, which is typically about 5000 points in length for a 2 second audio clip. The graph display updates as the user adjusts the position; upon pressing "Save" the offset is converted to time in seconds and a new audio excerpt is extracted according to the new parameters. Alternatively, "Edit Track Data" allows for manual specification of a bar start time.

Edit Track Data provides a means for user to enter their own key parameters, including as mentioned above Tempo and Bar start time. A third option "Beats in Bar" is included for the rare occasion that a track does not have a traditional 4/4 meter. Entering new parameters in this box also re-triggers the analysis process.

Actions available to the user for a selected track include bar playback, highlighting of peaks in the amplitude curve, and a primitive display of the feature vector. These tools are all intended to speed up the correction process

Similar Tracks can be generated instantly with respect to the relevant parameters selected on the interface. The callback function parses the specified weights and tempo range restriction. Once a list is generated, highlighting any track in the display box plots an overlay of the selected track's envelope data over the amplitude data. A user can also load a track as active from this box.

5 Evaluation & Results

As others have noted (1; 10), establishing consistent musical ground truths tends to be the main obstacle to truly meaningful content-based analysis and categorization. The inability to objectively analyse such a system's performance can be attributed partly to a dearth of fully validated tempo and fingerprint information, and partly to fuzziness of class boundaries, i.e. contention over what exactly constitutes a specific style or sub-genre. The evaluation process for this project utilises a combination of objective, fully automated tests, and semi-automated subjective tests.

Efficient evaluation of the constituent algorithms' performance is greatly facilitated by the overarching system framework. Matlab scripts read analysis files from disk storage and output relevant information to a .csv file that is then subjectively evaluated against known track information. In the case of evaluating the quality of more advanced features, the user interface provides an intuitive means to completing an otherwise arduous task.

5.1 Dataset

As with any content-based analysis study, the choice of dataset is guaranteed to have a profound impact on the results of the investigation. For this project, great care was taken to curate a set of songs that accurately represented a number of well-known genres.

Genre	Electro	Footwork	House	Jungle	Techno	Garage	**Total**
Count	19	22	24	23	21	22	**131**

The theory behind selection of tracks was to exclude more experimental works that have significantly blurred genre boundaries. Inevitably there is still some overlap between genres even in human categorization terms, most notably House & Techno.

5.2 Tempo

The most fundamental rhythmic characteristic and inevitably the cornerstone of all other algorithms in the system, Tempo is the necessary first factor for evaluation. Truly accurate tempo estimation is not knowledge freely available, and various algorithms are affected to varying extents by the presence of polyrhythms (25) as well as other artistic stylings.

Automatically generated spreadsheets containing output from the tempo estimation algorithms were updated manually to record accuracy against known tempos for the tracks. When evaluating the two algorithms individually, the candidate with the strongest confidence is used for comparison. For all detailed analyses, the estimated tempo is deemed to be correct if within 0.2BPM either side of the tempo generated by Pioneer's Rekordbox software. Results for genre subsets and for the dataset as a whole are presented below, and discussed in 6.1.

Algorithm	Correct	Total	Accuracy (%)	Mean Error (BPM)
Ellis	35	131	26.7	5.38
MIRTempo	73	131	55.7	13.2
Combined	92	131	70.23	4.612

Figure 5: *Margin of Error 0.2bpm*

Algorithm	Correct	Total	Accuracy (%)	Mean Error (BPM)
Ellis	93	131	71.1	7.075
MIRTempo	75	131	57.23	15.15
Combined	112	131	85.5	6.61

Figure 6: *Margin of Error 1bpm*

	Ellis			MIRTempo			Combined		
Genre	Acc	Err	Tmp	Acc	Err	Tmp	Acc	Err	Tmp
Electro	42.11	0.24	131.58	47.37	14.74	131.95	73.68	1.02	131.95
Footwork	4.55	3.26	159.57	72.72	5.71	159.99	81.81	0.29	160
House	60	1.6	121.95	97.67	0.07	121.84	93.33	1.09	121.84
Jungle	13.79	26.65	150	45.27	109.85	20.69	26.9	27.9	131.32
Techno	42.86	2.46	128.21	85.71	0.16	127.45	85.71	0.18	127.98
Garage	22.73	0.32	131.58	18.74	16.29	121.88	36.36	0.57	131.99

Figure 7: *Performance of each algorithm on each genre subset.* ***Acc*** *is accuracy in %.* ***Err*** *is the mean error in bpm from estimated tempo to actual.* ***Tmp*** *is the median tempo for each subset as predicted by the algorithm. MoE 0.2bpm*

5.3 Pattern Extraction

The next major step to be evaluated is the process of searching a track for and extracting a 1-bar long snippet that contains a recognisable and descriptive rhythmic pattern. The entire process consists of several algorithms, but to evaluate the output of each component procedure against user-specified values would be impractically time-consuming and not particularly informative.

Evaluation of this step is therefore carried out at a slightly higher-level, and gives an indication of the system's overall capability to identify locations of salient patterns in a track, given its correct tempo. A ground truth has to be determined somewhat subjectively for each track individually by assessing the suitability of the bar chosen for representation. The user interface is hugely helpful in this evaluation process, as it provides a means to sequentially assess each track visually and aurally; the interface plots the amplitude envelope and offers an option to play the bar from which it was extracted.

Additionally, for testing purposes, a simple command to display the generated feature vector is included. The pattern extraction was considered successful if no adjustment is required to produce a meaningful feature vector.

The results for each genre subset and the dataset as a whole are presented below, and discussed in 6.2:

Genre	Correct	Total	Accuracy (%)	Median Err (s)
Electro	15	19	78.95	0.054
Footwork	14	22	63.64	1.5
House	21	24	87.5	0.544
Jungle	12	23	52.17	9.54
Techno	18	21	85.71	0.072
Garage	12	22	54.55	5.927
Overall	92	131	70.23	1.5

Figure 8: *Accuracy of pattern extraction algorithm*

5.4 Content-Based Similarity

The ability (or inability) of the system to generate meaningful recommendations for a query track is in many ways its defining quality. This aspect of the system is evaluated with a KNN-based cross-classification technique. Track-to-track distances are computed using the Edit distance between each pair of corresponding frequency band vectors, with weights of 3, 1, 1 as discussed in section 4.3.2. The predicted genre for each query track is defined as the genre that occurs most frequently among its `k` most similar tracks. This method of evaluation is satisfactory as it concisely summarises the distribution of genres among track suggestions for the dataset as a whole.

The results of track-track distances generated under 3 distinct configurations are considered for classification analysis. Firstly, distances are produced immediately after the pattern extraction process - i.e. when no corrections have been made to the estimated bars. Next, the distances produced from a fully corrected set of patterns is considered. Finally, and somewhat conclusively, we analyse the suggestions provided from a fully corrected set of patterns and restrained by an 8% tempo filter.

The tables below show the accuracy of classification for each of the three configurations described above, each for a selection of values of `k` in between 1 and 20.

k	1	5	9	**13**	17	21
correct	50	57	58	**64**	62	60
accuracy	38.2	43.5	44.3	**48.9**	47.3	45.8

Figure 9: *Accuracy of classification for* ***uncorrected*** *data*

k	1	5	9	13	17	**21**
correct	59	67	68	66	68	**71**
accuracy	45	51.1	51.9	50.3	51.9	**54.2**

Figure 10: *Accuracy of classification for fully* ***corrected*** *data*

k	1	5	9	**13**	17	21
correct	76	85	89	**92**	87	82
accuracy	58	64.9	67.9	**70.2**	66.4	62.6

Figure 11: *Accuracy of classification for fully **corrected** data **w/ tempo** filter*

These results show that as expected, correctly positioned bars are significantly important. Although there appears to be a high number of incorrect predictions, understanding the causes and implications of incorrect predictions is key to evaluating the success of the system. Also emphasised is the impact of a tempo filter, the reasons for which can again be derived from detailed analysis of the results.

Distribution of predictions can be visualised in the confusion matrices shown below. In the matrices, rows correspond to actual annotated genres, and columns correspond to predicted genres. The counts indicate the number of tracks in each genre that were predicted as belonging to each other genre.

	Elc	Fwk	Hse	Jng	Tec	Grg	Total
Elc	**10**	4	1	3	1	0	19
Fwk	2	**13**	1	5	1	0	22
Hse	0	0	**22**	1	1	0	24
Jng	0	6	1	**10**	1	5	23
Tec	1	0	12	0	**8**	0	21
Grg	2	4	7	7	1	**1**	22

Figure 12: *Classification confusion matrix for **uncorrected** data*

	Elc	Fwk	Hse	Jng	Tec	Grg	Total
Elc	**9**	2	3	5	0	0	19
Fwk	0	**11**	3	8	0	0	22
Hse	0	0	**21**	0	3	0	24
Jng	0	1	1	**21**	0	0	23
Tec	0	0	13	0	**8**	0	21
Grg	1	1	3	17	0	**0**	22

Figure 13: *Classification confusion matrix for fully* ***corrected*** *data*

	Elc	Fwk	Hse	Jng	Tec	Grg	Total
Elc	**8**	0	1	2	0	8	19
Fwk	0	**16**	0	6	0	0	22
Hse	0	0	**22**	0	2	0	24
Jng	0	2	0	**21**	0	0	23
Tec	0	0	13	0	**8**	0	21
Grg	5	0	0	0	0	**17**	22

Figure 14: *Classification confusion matrix for fully* ***corrected*** *data* ***w/ tempo*** *filter*

6 Discussion

This chapter discusses the results presented in section 5.

6.1 Tempo

From a statistical analysis of the various tempo estimations (5.2), we can note a number of things.

Firstly, whilst the Ellis algorithm scores very low accuracies for most genres, the mean error is generally also very low. This indicates that the tempo estimations are very close to the actual values, but the narrow margin of error used means they are recorded as failures. This can be verified by increasing the correctness threshold to 1BPM, which drastically increases the overall accuracy from 26.7% to 71.1%. It is in large part this property of Ellis' algorithm that makes it ideal to 'correct' the values produced by `mirtempo`.

`Mirtempo` is considerably more computationally expensive than the Ellis algorithm, and as a result produces far more precise tempo estimations, as reflected in the general accuracy increase over Ellis. However, `mirtempo` often struggles with the trickier break-beat genres, scoring a low accuracy and high mean error, indicating that many estimations are wildly inaccurate. The cause of this is an issue at the heart of the tempo estimation problem, which is the difficulty introduced by polyrhythms. To help understand this, visualize the spread of results produced for the Jungle subset.

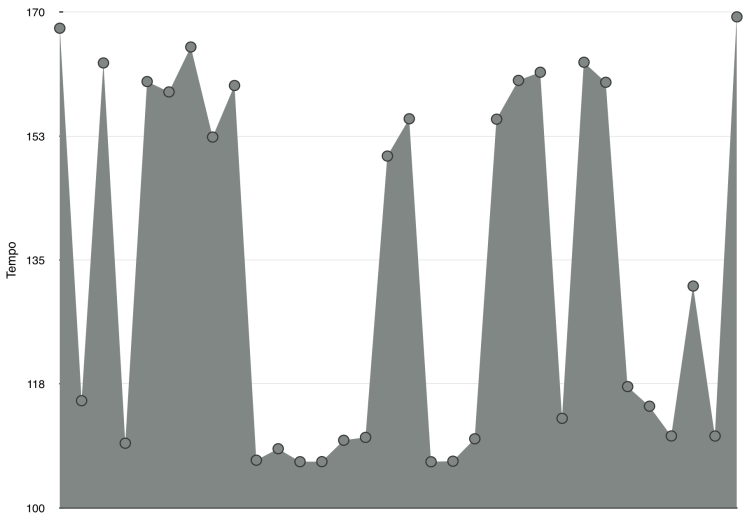


Figure 15: *Estimated tempo of each track in the Jungle subset*

The graph shows that almost all tempo estimations fall within one of the two ranges 110-118 or 150-170bpm. This confirms that the error is down to ambiguity of interpretation of rhythmic patterns, as the ratio between the actual and estimated tempo of all tracks in the lower, incorrect sector is within a very small range of 1.5. The off-kilter breakbeat patterns synonymous with jungle music introduce algorithmic ambiguities that are extremely difficult for algorithms to resolve, as is demonstrated by the similarly poor performance by the two separate tempo estimation approaches evaluated. This same license of percussive expression introduces issues with Garage music, although to a lesser extent.

To somewhat ameliorate this flaw, the combined approach works by using the vaguely accurate but imprecise estimations of Ellis' algorithm to pick a best candidate from those offered by `mirtempo`. The evaluation shows that this produces notably superior performance compared to the each algorithm separately, and therefore is the default choice of tempo estimation in the system.

6.2 Pattern Extraction

The inaccurate results from the pattern extraction algorithm (5.3) can be categorized into two distinct groups - bars extracted with incorrect phase information, and bars extracted from unrepresentative regions of tracks. Many tracks have beat-less sections containing low frequency components and these are regularly treated by the algorithm as candidates. Bars extracted from these areas required large corrections ranging from 2-60 seconds, and

the median error for each genre indicates that these issues were most prevalent in Garage and Jungle music, the two styles most partial to bassline-containing breakdowns.

Most inaccuracies required significantly smaller adjustments anywhere in the region of 25ms to 2 seconds, and could be introduced by errors in several of the other constituent algorithms. For example, an incorrect tempo for a track results in incorrect segmentation and all of its bars are consequently incorrect length. This issue can be somewhat mitigated by correcting the tempos before, although even a discrepancy of 0.3bpm (which would not obvious to a user) can be known to cause issues with the segmentation algorithm.

Inconsistencies across tracks are inevitably the root cause of many errors. When algorithmically locating the position in the bar of a kick-drum, for example, consideration had to be made for the length of the attack onset phase, which tended to differ across songs that used varying drum samples. This value can be computed from analysis of the onset curve output by `mironsets`, and so in this project is calculated and stored for each track individually at the downbeat detection phase.

6.3 Content-Based Similarity

Analysis of the figures presented in 5.4 reveals a number of important facts regarding both the performance of the system and the relationships between pieces of music across genres.

We can immediately conclude that the descriptor chosen to represent each track is indeed capable to some extent of encoding important rhythmic information and subsequently ranking and grouping collections of electronic music. This is particularly demonstrated by the analysis presented in figures 10 and 11, where the overall classification precision is upwards of 50%. This represents a significant increase over the random prediction value which would be around 16 %. In addition, tracks from the more distinctive genres can be classified with success of over 90%. To this end, particularly when filtering suggestions through a permitted tempo range, the categorization element of the project can be considered successful.

The categorization capabilities of any content-based recommendation system that utilises such a small range of features from the data will however always be limited. The rhythmic profile introduced in this project is in some cases perfectly distinctive and in others

completely unable to separate genres. This is best illustrated by the Garage subset, in particular its interactions with the other break-beat genres, Electro and Jungle. Tracks within these three genres are all found to contain highly similar rhythmic profiles, an example of which is displayed below.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
low	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
mid	1	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-
high	-	-	1	-	-	-	1	-	1	-	-	-	1	-	1	-

Figure 16: *Vector representation of a rhythm shared by Garage, Jungle, Electro tracks, as well as other tracks from genres outside the scope of this project*

The irregular kick pattern presented here, and a number of others like it are found in many tracks across various genres and will inevitably introduce issues when assessing straight-forward genre classification. The confusion matrix reflects this - when tempo restrictions are not applied, Garage tracks are regularly matched with Electro and Jungle tracks. With a tempo filter included, Garage and Electro tracks form a definitive cluster, but Jungle is generally left out due to its significantly different tempo.

Some basic familiarity with the styles analysed here is therefore knowledge enough to state that it is highly unlikely that such a system would be able to fully distinguish between Garage and Electro, as the two styles not only exhibit highly similar inner and outer metrical qualities, but also tend to fall within the same tempo range. This same phenomenon can be observed with the confusion between House and Techno.

It is important to note these attributes of the dataset when evaluating the performance of the system. Taking this into consideration helps to portray the accuracy of suggestions in a clearer light, as there is very little overlap between genres that are not generally known to share rhythmic characteristics. Indeed, one of the main goals of the project was to provide a tool capable of genre-ignorant, rhythmically-similar suggestions, rather than a tool intended for categorization. In this aspect, the software produced absolutely fulfils its aims.

7 Conclusions & Future Work

The proposed feature representation shows real promise as a low-dimensional yet effective descriptor, at least for the subset in question. This project clarifies the important considerations involved in generating such vectors, and implements a number of approaches to the various problems. The results are reasonably successful, but there is undoubtedly room for improvement at several stages of the process.

Tempo, Segmentation, Pattern Extraction

Access to state of the art tempo estimation, such as that packaged with many existing DJ applications, would have significant implications for the system's performance. In particular, the inability to compute tempo and downbeats with perfect accuracy meant that certain tracks had incorrect bar delimitations and phase information, representing a notable issue for the system. To clarify, although a computer can usually calculate to an acceptable degree of accuracy the location and periodicity of beats, it is far more complicated to algorithmically determine on which of these beats does the start of a bar lie. The degree of severity of these innacuracies ranged from a few milliseconds to entirely nonsensical bar suggestions. Even a phase-start inaccuracy of 100ms (i.e. around 1/16 of a bar length) can have significant implications on the validity of a feature extracted from this bar length pattern. Improvements to this process would clearly be reflected in the overall outcome, but significant research has been dedicated to this problem and it remains unsolved, so the approach chosen here is satisfactory.

A potential for improvement to the segmentation process is an approach similar to that adopted in (6), whereby the start time of every bar is determined by searching for points within an expected range that have maximum correlation with the original bar time. This would allow for a slightly higher error in the tempo estimation whilst retaining performance, as the system would intelligently pick each bar length.

Frequency Decomposition

The frequency decomposition specified in (17) and implemented in (12) is thought to be representative of auditory perception, and so should be wholly appropriate for this investigation. However, an interesting route for a more intelligent and flexible approach to this problem is detailed in (16). The signal is first decomposed into 24 bark bands, before a novelty function is applied to a loudness measure on each of these bands, using

the detected points of change to define boundaries by which to group rhythmic elements. This approach was found to be particularly effective for electronic music by "not imposing any constraints on the possible instrument sounds that contribute to the characteristic rhythmic pattern".

Vector Representation

A number of checks and adjustments are incorporated to increase robustness when generating a vector from an amplitude envelope, but could still be improved upon. A more intelligent algorithm could recognise when the vector has been mispositioned due to an incorrect offset in the pattern extraction stage, and adjust the resultant vector accordingly. This would be very difficult to implement due to natural inconsistencies introduced by stylistic differences between songs, but done correctly would significantly reduce the need for correction of bar positioning.

The complexity of the vector itself could also be increased. For example, each peak could be represented in the vector in terms of its height, rather than just its presence. Such a representation would not only require a more sophisticated and expensive similarity measure, but would also complicate the problem of distinguishing between attack events and general music events. It is possible however that something like this could offer an improvement to the performance of the conceptually simple vectors utilised in this project.

Interface

There are several enhancements that could be made to the interface. An interesting option would be to allow a user to specify a rhythm to be used for content-based retrieval, either entered manually or recognised from tapping keys. An interactive graph display rather than the dumb one used here would further ease the correction process. Also, an automated option to import meta-data such as tempo or even beat times would increase the effectiveness and simplicity of the software.

Not only does the software developed throughout this project present an effective and meaningful way to rhythmically fingerprint pieces of electronic music, but the framework and interface provided also offer an intuitive means for any future researcher to improve, implement and test any of the defining operations.

8 Appendix

.1 Project Brief

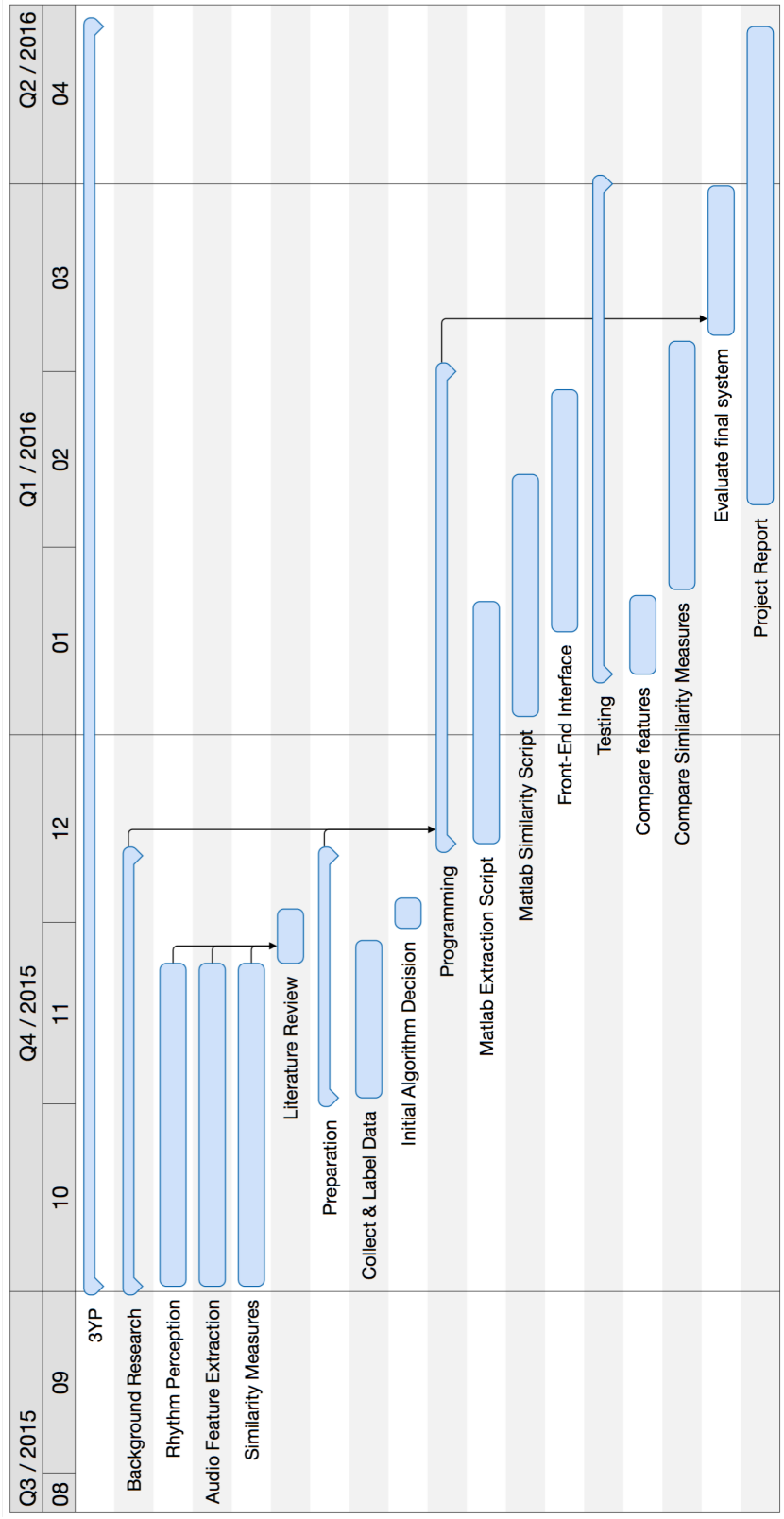
AUTOMATIC SUB-GENRE CLASSIFICATION OF ELECTRONIC MUSIC BASED ON RHYTHMIC COMPONENTS AND CHARACTERISTSICS

The vast majority of modern electronic music played by DJs today can be categorised into one of several sub-genres, or styles, for example; House, Garage, Dubstep or Jungle, each of which can be almost scientifically defined by a combination of its tempo (bpm) and its rhythmic characteristics and patterns. Whilst most work into audio classification and identification focuses on analysing tonal changes between segments, this project will attempt to classify works by extraction and analysis of its rhythmic elements, how they are grouped, and how often each individual drum hit (or any other significant rhythmic element) occurs, and with what velocity.

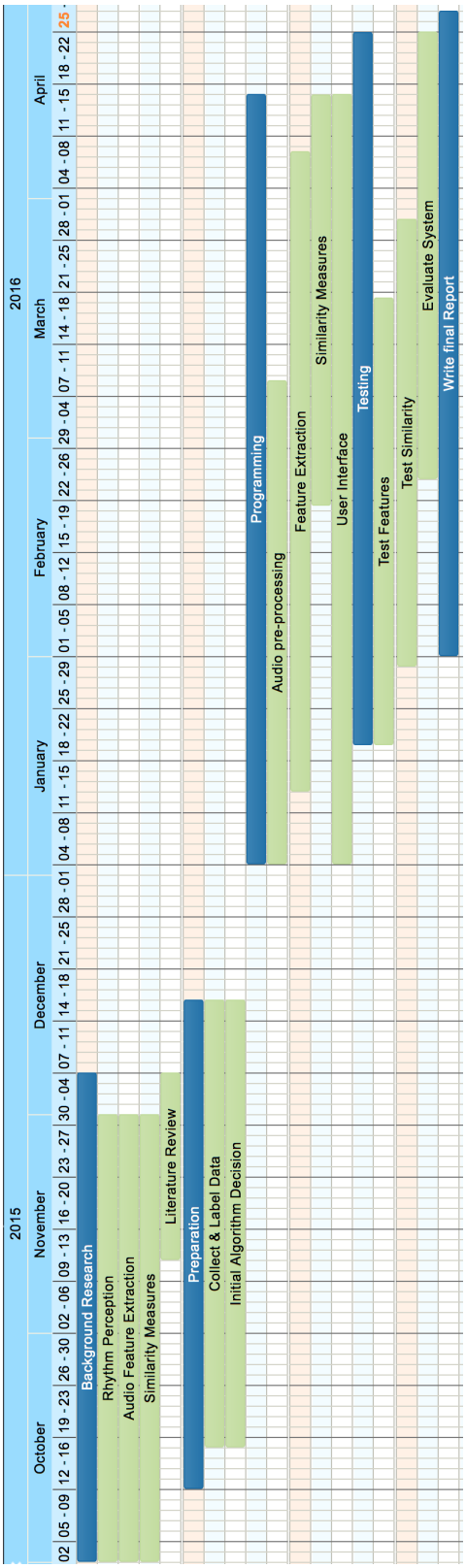
This approach has been explored somewhat in Simon Dixon et. al.'s 2004 paper '*TOWARDS CLASSIFICATION OF MUSIC VIA RHYTHMIC PATTERNS'*, whose work showed moderate success when applied to standard and Latin ballroom dance works. By adopting a similar methodology I will aim to achieve equivalent, if not better results on large libraries of contemporary dance music. Splitting the input single into its 3 main frequency bands (low, mid, hi) will enable the isolation and analysis of specific instruments that are most integral to defining the song's style. This technique was not explored but was mentioned by Dixon et. al - in theory the nature of the music being analysed in this case should lend it more relevance.

The resulting software will have real-world use in grouping potentially huge libraries of electronic music into collections or playlists based on their specific style and intensity. Additionally, a similarity index would give a user the ability to see any given track's closest rhythmic neighbours, a tool that would prove useful for a DJ desiring near-undetectable blends between songs. Finally, by combining the results of the proposed classification system with those of an existing key detection software, the tool would be able to detect finer nuances between sub-genres, increasing its utility. The software will be supported by an interface based on a simple conceptual model that is familiar to those with large digital collections.

.2 Planned Progress (GANTT)



.3 Actual Progress (GANTT)



References

[1] Bogdanov, D., Serrà, J., Wack, N., and Herrera, P. (2009). From Low-Level to High-Level: Comparative Study of Music Similarity Measures. *2009 11th IEEE International Symposium on Multimedia*, 7(3):453–458.

[2] Chew, E., Volk, A., and Lee, C.-Y. (2005). Dance Music Classification Using Inner Metric Analysis. In *The Next Wave in Computing, Optimization, and Decision Technologies*, number October 2015, pages 355–370. Springer US.

[3] Cooper, G. and Meyer, L. B. (1960). *The Rhythmic Structure of Music*. The Univ. of Chicago Press, Chicago.

[4] Dixon, S. (2001). An Interactive Beat Tracking and Visualisation System The Audio-Graphical User Interface.

[5] Dixon, S. (2006). MIREX 2006 Audio Beat Tracking Evaluation : BeatRoot.

[6] Dixon, S., Gouyon, F., and Widmer, G. (2004). Towards characterisation of music via rhythmic patterns. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 5:509–516.

[7] Ellis, D. P. W. (2007). Beat Tracking by Dynamic Programming. *Journal of New Music Research*, 36(1):51–60.

[8] Foote, J. and Uchihashi, S. (2001). The beat spectrum: a new approach to rhythm analysis. *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, pages 881 – 884.

[9] Gabrielsson, A. (1973). Similarity ratings and dimension analyses of auditory rhythm patterns. *Scandinavian journal of psychology*, 14(4):244–260.

[10] Gouyon, F. and Herrera, P. (2003). Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptions. *AES 114th Convention*, (1960).

[11] Honing, H. (2002). Structure and interpretation of rhythm and timing. *Tijdschrift voor Muziektheorie*, 16(1998).

[12] Lartillot, O. and Toiviainen, P. (2007). A matlab toolbox for musical feature extraction from audio. *International Conference on Digital Audio ...*, pages 1–8.

[13] Mckinney, M. M. F. and Breebaart, J. (2003). Features for Audio and Music Classification. *Proc ISMIR*, 4:151–158.

[14] Nestke, A. and Noll, T. (2002). Inner Metric Analysis. pages 1–20.

[15] Pampalk, E., Rauber, A., and Merkl, D. (2002). Content-based organization and visualization of music archives. *Proceedings of the tenth ACM international conference on Multimedia - MULTIMEDIA '02*, page 570.

[16] Panteli, M. (2014). Modeling Rhythm Similarity for Electronic Dance Music. (Ismir):537–542.

[17] Patterson, R. D., Robinson, K., Holdsworth, J., McKeown, D., Zhang, C., and Allerhand, M. (1992). Complex sounds and auditory images. *Simulation*, 83(1992):429–446.

[18] Paulus, J. and Klapuri, A. (2002). Measuring the Similarity of Rhythmic Patterns. *Signal Processing*, 1:44.

[19] Rocha, B., Bogaards, N., and Honingh, A. (2013). Segmentation and Timbre Similarity in Electronic Dance Music. (1).

[20] Smith, L. M. (2010). Rhythmic similarity using metrical profile matching. pages 177–182.

[21] Toussaint, G. (2004). A comparison of rhythmic similarity measures. *Proc. International Conference on Music Information Retrieval (ISMIR 2004)*, pages 242–245.

[22] Tzanetakis, G., Essl, G., and Cook, P. (2001). Automatic musical genre classification of audio signals. *Proceedings of the Second International Symposium on Music Information Retrieval*, page 6 total pages.

[23] Wang, A. (2006). The Shazam music recognition service. *Communications of the ACM*, 49(8):44.

[24] Welsh, M., Borishov, N., Hill, J., von Behren, R., and Woo, A. (1999). Querying large collections of music for similarity. *UC Berkeley Technical Report*, (UCB/CSD-00-1096).

[25] Whiteley, N., Cemgil, A. T., and Godsill, S. (2006). Sequential Inference of Rhythmic Structure in Music Audio. (2):1–4.

[26] Zapata, J. R. and Gómez, E. (2011). Comparative evaluation and Combination of Audio Tempo Estimation Approaches. *AES 42nd International Conference*, pages 1–10.