**Faculty of Mathematics and Computer Sciences**



**Master’s Thesis**

**Music Similarity Analysis Using the Big Data Framework Spark**

|  |  |
| --- | --- |
| presented by: | Johannes Schoder |
| born: | 03. September 1994 |
| ID: | 169197 |
| course of studies: | M.Sc. Informatik |
| supervisors: | Prof. Dr. Martin Bucker, Ralf Seidler¨ |

**Contents**

[**Abbreviations** **iv**](#_Toc142613)

[**List of Figures** **v**](#_Toc142614)

[**List of Tables** **vii**](#_Toc142615)

[**List of Code Snippets** **viii**](#_Toc142616)

[**1 Introduction** **1**](#_Toc142617)

[1.1 Why and how? 1](#_Toc142618)

[1.2 Overview 2](#_Toc142619)

[1.3 Conclusions 3](#_Toc142620)

[**2 Music Information Retrieval and Big Data** **4**](#_Toc142621)

[2.1 Audio Features 4](#_Toc142622)

[2.1.1 Fourier Transformation 4](#_Toc142623)

[2.1.2 MFCC 5](#_Toc142624)

[2.1.3 other audio features 8](#_Toc142625)

[2.2 MIR Toolkits 9](#_Toc142626)

[2.2.1 Low-level audio feature extraction 9](#_Toc142627)

[2.2.2 Music Similarity 9](#_Toc142628)

[2.2.3 Melody/ pitch extraction 10](#_Toc142629)

[2.3 Music Similarity Measurements 12](#_Toc142630)

[2.3.1 Timbre based 12](#_Toc142631)

[2.3.2 Pitch based 12](#_Toc142632)

[2.3.3 Rhythm based 12](#_Toc142633)

[2.3.4 Metadata based/ Collaborative Filtering 12](#_Toc142634)

[2.3.5 Note based 13](#_Toc142635)

[2.3.6 Genre specific features 13](#_Toc142636)

[2.3.7 summary 14](#_Toc142637)

[2.4 Data aggregation 14](#_Toc142638)

[2.4.1 Datasets 14](#_Toc142639)

[2.4.2 Alternatives 16](#_Toc142640)

[2.5 Big Data 19](#_Toc142641)

[2.5.1 Hadoop 20](#_Toc142642)

[2.5.2 Spark 21](#_Toc142643)

[2.5.3 Music Similarity with Big Data Frameworks 26](#_Toc142644)

[**3 Similarity Analysis** **27**](#_Toc142645)

[3.1 Timbre Similarity 27](#_Toc142646)

[3.1.1 Euclidean Distance 27](#_Toc142647)

[3.1.2 Single Gaussian Model 28](#_Toc142648)

[3.1.3 Gaussian Mixture Models and block-level features 29](#_Toc142649)

[3.1.4 Validation 29](#_Toc142650)

[3.2 Melodic Similarity 32](#_Toc142651)

[3.2.1 Representation 32](#_Toc142652)

[3.2.2 Chroma Features pre-processing 32](#_Toc142653)

[3.2.3 Similarity of chroma features 35](#_Toc142654)

[3.2.4 Validation 41](#_Toc142655)

[3.3 Rhythmic Similarity 41](#_Toc142656)

[3.3.1 Beat histogram 41](#_Toc142657)

[3.3.2 Rhythm patterns 42](#_Toc142658)

[3.3.3 Rhythm Histogram 44](#_Toc142659)

[3.3.4 cross-correlation 44](#_Toc142660)

[3.4 Summary 45](#_Toc142661)

[3.4.1 Timbre Similarity 45](#_Toc142662)

[3.4.2 Melodic Similarity 46](#_Toc142663)

[3.4.3 Rhythm Similarity 46](#_Toc142664)

[3.4.4 Feature files 47](#_Toc142665)

[**4 Implementation** **48**](#_Toc142666)

[4.1 Audio Feature Extraction 48](#_Toc142667)

[4.1.1 Test Datasets 48](#_Toc142668)

[4.1.2 Feature Extraction Performance 49](#_Toc142669)

[4.2 Big Data Framework Spark 55](#_Toc142670)

[4.2.1 Underlying hardware 55](#_Toc142671)

[4.2.2 Workflow 56](#_Toc142672)

[4.2.3 Data preparation 56](#_Toc142673)

[4.2.4 Distance Computation 58](#_Toc142674)

[4.2.5 distance scaling 66](#_Toc142675)

[4.2.6 Combining different measurements 67](#_Toc142676)

[4.2.7 performance 68](#_Toc142677)

[4.2.8 possible improvements and additions 76](#_Toc142678)

[**5 Results** **78**](#_Toc142679)

[5.1 Objective evaluation 78](#_Toc142680)

[5.1.1 feature correlation and distance distribution 78](#_Toc142681)

[5.1.2 Cover song identification 83](#_Toc142682)

[5.1.3 Genre similarity 85](#_Toc142683)

[5.1.4 Rhythm Features 88](#_Toc142684)

[5.2 Subjective evaluation 89](#_Toc142685)

[5.2.1 Beyond genre boundaries 89](#_Toc142686)

[5.2.2 personal music taste 90](#_Toc142687)

[**6 Summary** **91**](#_Toc142688)

[6.1 Conclusion 91](#_Toc142689)

[6.2 Outlook 92](#_Toc142690)

[**References** **93**](#_Toc142691)

[**7 Appendix** **99**](#_Toc142692)

[7.1 Feature Analysis 99](#_Toc142693)

[7.2 Spotipy Data Miner 100](#_Toc142694)

[7.3 Contend of the CD 102](#_Toc142695)

# Abbreviations

**BH** beat histogram

**BPM** beats per minute

**BRP** bucketed random projection

**DAG** directed acyclic graph

**DCT** discrete cosine transformation

**DF** Spark DataFrame

**DTW** dynamic time warping

**ESA** explicit semantic analysis

**FFT** fast fourier transformation

**GMM** gaussian mixture model

**HDFS** hadoop distributed file system

**HT** hyperthreading

**JS divergence** Jensen Shannon

**JVM** Java virtual machine

**KL divergence** Kullback-Leibler divergence

**LSH** Locality-sensitive hashing

**MFCC** mel frequency cepstral coefficients

**MIDI** musical instrument digital interface

**MIR** music information retrieval

**MP** mutual proximity

**MSD** million song dataset

**RDD** resilient distributed dataset

**RH** rhythm histogram

**RP** rhythm pattern

**SKL** symmetric Kullback-Leibler divergence

**SQL** structured query language

**UDF** user defined function

**YARN** yet another resource negotiator

# List of Figures

|  |  |
| --- | --- |
| 2.1 Frequency Space . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| 2.2 Sweep Sinal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| 2.3 MFCCs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| 2.4 Timbre guitar vs. piano . . . . . . . . . . . . . . . . . . . . . . . . . . | 7 |
| 2.5 MFCC statistics guitar vs. piano . . . . . . . . . . . . . . . . . . . . . | 7 |
| 2.6 MFCC statistics guitar vs. piano . . . . . . . . . . . . . . . . . . . . . | 8 |
| 2.7 Features of the song Layla by Eric Clapton . . . . . . . . . . . . . . . . | 8 |
| 2.8 Features of the song Layla by Eric Clapton . . . . . . . . . . . . . . . . | 9 |
| 2.9 Original Scores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| 2.10 Aubio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| 2.11 Melodia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| 2.12 Transcription . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| 2.13 Datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| 2.14 Spotify API . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 17 |
| 2.15 million song dataset genre distribution . . . . . . . . . . . . . . . . . . | 19 |
| 2.16 MapReduce [57] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 21 |
| 2.17 Spark Cluster . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 22 |
| 2.18 Spark Application UI . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 23 |
| 3.1 Construction Noise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 30 |
| 3.2 Chroma Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 32 |
| 3.3 Bandpass - Sia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 33 |
| 3.4 Thresholded Chroma Features - Sia . . . . . . . . . . . . . . . . . . . . | 34 |
| 3.5 Processed Chroma Features - Sia . . . . . . . . . . . . . . . . . . . . . | 34 |
| 3.6 Workflow chroma feature extraction . . . . . . . . . . . . . . . . . . . . | 35 |
| 3.7 Processing Step 3 Chroma Features . . . . . . . . . . . . . . . . . . . . | 35 |
| 3.8 cross-correlation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 38 |
| 3.9 beat-aligned chromagram . . . . . . . . . . . . . . . . . . . . . . . . . . | 39 |
| 3.10 Cross-correlation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 40 |
| 3.11 Cross-correlation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 40 |

|  |  |
| --- | --- |
| 3.12 Cross-correlation filtered . . . . . . . . . . . . . . . . . . . . . . . . . . | 41 |
| 3.13 Beat Histogram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 42 |
| 3.14 Rhythmic Patterns . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 43 |
| 3.15 Rhythm Pattern extraction [71] . . . . . . . . . . . . . . . . . . . . . . | 43 |
| 3.16 Rhythm Histogram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 44 |
| 3.17 Detected Onsets (first 30 seconds) . . . . . . . . . . . . . . . . . . . . . | 45 |
| 4.1 Performance of various toolkits on a single computer . . . . . . . . . . | 53 |
| 4.2 File sizes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 55 |
| 4.3 Workflow Spark . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 56 |
| 4.4 Lazy evaluation and caching . . . . . . . . . . . . . . . . . . . . . . . . | 66 |
| 4.5 #Executors spawned . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 69 |
| 4.6 Performance on different feature types . . . . . . . . . . . . . . . . . . | 70 |
| 4.7 Performance ARA, full workload, (MFCC + Notes + RP) . . . . . . . | 72 |
| 4.8 Performance ARA, full workload, (JS + Chroma + RP) . . . . . . . . | 72 |
| 4.9 Workflow Merged DF . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 72 |
| 4.10 two subsequent songs, all features . . . . . . . . . . . . . . . . . . . . . | 73 |
| 4.11 Descending importance filter and refine, all features . . . . . . . . . . . | 74 |
| 4.12 Performance / Executors (36 CPU cores each) . . . . . . . . . . . . . . | 76 |
| 5.1 Feature space example . . . . . . . . . . . . . . . . . . . . . . . . . . . | 79 |
| 5.2 correlation 95 songs, 19 genres (5 each), 1517 artists . . . . . . . . . . . | 79 |
| 5.3 Cumulative distributions . . . . . . . . . . . . . . . . . . . . . . . . . . | 80 |
| 5.4 Correlation of features depending on SKL scaling . . . . . . . . . . . . | 81 |
| 5.5 Correlation of features . . . . . . . . . . . . . . . . . . . . . . . . . . . | 81 |
| 5.6 correlation 95 songs, 19 genres (5 each), 1517 artists . . . . . . . . . . . | 82 |
| 5.7 genre recall . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 85 |
| 5.8 distances 1 song, Rock&Pop, 1517 artists, 4 genres . . . . . . . . . . . | 86 |
| 5.9 distances 1 song, electronic, 1517 artists, 4 genres . . . . . . . . . . . . | 87 |
| 5.10 Rhythm features/ BPM . . . . . . . . . . . . . . . . . . . . . . . . . . | 88 |
| 7.1 distances 1 song (Soundtrack), 5 genres (10 songs each) . . . . . . . . . | 99 |

# List of Tables

2.1 music datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

4.1 appropriate music datasets . . . . . . . . . . . . . . . . . . . . . . . . . 49

5.1 Cover song recognition - Top 1 . . . . . . . . . . . . . . . . . . . . . . . 83

5.2 Cover song recognition - Top 5 . . . . . . . . . . . . . . . . . . . . . . . 83

# List of Code Snippets

|  |  |
| --- | --- |
| 2.1 MIR Toolkit Similarity . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| 2.2 example cluster configuration python . . . . . . . . . . . . . . . . . . . | 24 |
| 2.3 lazy evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 25 |
| 4.1 librosa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 49 |
| 4.2 essentia standard . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 50 |
| 4.3 essentia streaming . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 51 |
| 4.4 parallel python . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 52 |
| 4.5 mpi4py . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 54 |
| 4.6 notes preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 57 |
| 4.7 rp preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 57 |
| 4.8 euclidean distance DF . . . . . . . . . . . . . . . . . . . . . . . . . . . | 58 |
| 4.9 Filter for requested song . . . . . . . . . . . . . . . . . . . . . . . . . . | 59 |
| 4.10 euclidean distance RDD . . . . . . . . . . . . . . . . . . . . . . . . . . | 59 |
| 4.11 bucketed random projection . . . . . . . . . . . . . . . . . . . . . . . . | 60 |
| 4.12 cross-correlation scipy . . . . . . . . . . . . . . . . . . . . . . . . . . . | 61 |
| 4.13 cross-correlation numpy . . . . . . . . . . . . . . . . . . . . . . . . . . | 61 |
| 4.14 Jensen-Shannon-like Divergence . . . . . . . . . . . . . . . . . . . . . . | 62 |
| 4.15 Kullback-Leibler Divergence . . . . . . . . . . . . . . . . . . . . . . . . | 63 |
| 4.16 Levenshtein DataFrame . . . . . . . . . . . . . . . . . . . . . . . . . . . | 64 |
| 4.17 Levenshtein RDD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 64 |
| 4.18 Spark lazy evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 65 |
| 4.19 Minimum and maximum aggregation separate . . . . . . . . . . . . . . | 67 |
| 4.20 Minimum and maximum aggregation optimized . . . . . . . . . . . . . | 67 |
| 4.21 cluster setup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 68 |

**Abstract**

**Estimating music similarity by merging various similarity measurements with the help of the Big Data Framework Spark.**

This thesis is about the comparison of construction noise and modern day music. The field of music information retrieval (MIR) in computer science is mostly a data driven and purely mathematical topic. The goal of this thesis is to merge the fields of computer science with music theoretical knowledge and to find potential weak spots of current music similarity algorithms focusing on single aspects like melody, rhythm and timbre.

# 1. Introduction

The idea originated from Dr. T. Bosse from the chair for advanced computing at the Friedrich-Schiller-University (FSU) in Jena. When proposing the idea for a master thesis with the topic of ”Music similarity measurement using genre specific features” using different guitar play styles in modern day metal music, he jokingly said that he would also like to know how metal music compares to construction building noise. The idea is actually not so groundless, considering that most people would agree on the fact that metal music is often described as noise by people not used to listening to genres like death and black metal. This thesis is meant to evaluate how music similarity algorithms compare construction noise to common musical genres and how to possibly improve these algorithms.

## 1.1 Why and how?

Why is music similarity research even necessary?

First of all, there is no fixed definition of music similarity so far. This is one of the first problems, dealing with music similarity. This topic offers multifaceted approaches. Merging multiple approaches with different weights can offer a more diverse music recommendation system. To do this, a lot of different data is required.

Content (music features) and context (listener behaviour) data can be fed into a big data framework to speed up operations. Collecting this data for large amounts of songs results in big datasets that need to be explored efficiently.

What to improve?

”[...] Spotify Radio, iTunes Radio, Google Play Access All Areas and Xbox Music. Recommendations are typically made using (undisclosed) content-based retrieval techniques, collaborative filtering data or a combination thereof.” [1, p. 9] The goal of this work is, to propose a transparent music similarity retrieval method based on various weighted contextual and content-based data. Applying different weights to different features allows similarity retrieval methods to search for different kind of similarity. E.g. weighing the tempo and beat of a song more then melodic similarity allows the creation of playlists for workout and sport, whilst melodic/ timbre etc. similarities allows to search for similar songs from musical subgenres. The user would get to decide what kind of playlist he wants to create. Adding contextual data and feeding it to an algorithm could add more or less popular music to the playlist with the goal to discover new upcoming artists or get other popular and most listened music. For this thesis however, the focus lies on content-based data/ audio features.

How to approach this?

First of all, a lot of data is required. In the first part, different scientific datasets are evaluated. Secondly, the available features are shown and explained. In the third part, different metrics are explained using the previously explored features. Lastly, a big data approach to efficiently use the gathered features is proposed and evaluated. A way to evaluate the results is also proposed.

## 1.2 Overview

*Structure:*



PARTIV:Per-

formanceTuning



PARTIII:Simi-

larityEstimation



PARTII:FeatureExtraction



PARTI:DataAggregation

## 1.3 Conclusions

Why using a big data framework would help: music similarity is not well defined. It is a rather subjective value that differs from listener to listener. Two tracks could be considered as ”similar” when they are equal in tempo, loudness, melody, instrumentation, key, rhythm mood, lyrics or a combination of more than a few of these features. The usage of a big data framework allows to create a variable/ fuzzy metric definition. Various parameters could easily be taken into consideration when calculating the musical distance of two different pieces. Using a Big Data framework, the problem of the fuzzy definition of music similarity could be avoided, if a metric can be found, that takes multiple of the accounted features of this thesis into consideration. Available information includes metadata, user data, audio features, sheet music and more.

*The idea of using genre specific features could be evaluated any further*

*Another important question is how to measure similarity algorithms. There are a few possibilities like genre, composer/ interpret or cover song identification. Or actual user data. MSD Challenge Dataset usable?[2]*

# 2. Music Information Retrieval and Big Data

The field of music information retrieval (MIR) is a large research area combining studies in computer science like signal processing and machine learning as well as psychology and academic music study. To get started, a brief overview is given in the next section providing the most important information about publicly available datasets, MIR toolkits and approaches to music similarity. Various datasets are presented and an overview over Big Data frameworks is given as well.

## 2.1 Audio Features

This section provides an overview about different music similarity measurements, audio features and metrics.

More in-depth information about the different metrics is given in chapter 3.1, 3.2 and

3.3

### 2.1.1 Fourier Transformation

Most of the algorithms start with switching from the time domain to the frequency domain, by performing a Discrete Fourier transform as described in equation 2.1 and

then compute the power spectrum (equation 2.2)

*K−*1

*Xm* = X *xk · e−*2*·Kπ·i·k·m* (2.1)

*k*=0

q

*|Xm|* = *Re*(*Xm*)2 + *Im*(*Xm*)2 (2.2)

Figure 2.1a shows the spectrogram (spectrum of frequencies over time) of the first bars of the song Layla by Eric Clapton. The sound sample was recorded on an electric guitar. Due to the fact that the human ear perceives sound in a non-linear matter, a logarithmic or Mel-scale is better to represent different pitches.

For example the note A4 is perceived at a frequency of 440Hz, the A note of next octave

(A5) is at 880Hz and the next one is at 1600Hz and so on. The Mel-scale [1, pp. 53f] was introduced to resemble the human perception of frequency (equation 2.3)

*f*

*m* = 1127 *· ln*(1 + ) (2.3)

700

The following plots were created with the librosa library [3]. The high dimensionality of

|  |  |
| --- | --- |
|  |  |

(a) Spectrogram (b) log scaled spectrogram

Figure 2.1: Frequency Space

the data is a problem for machine learning applications and music similarity tasks, as computation based on a vector with such a high dimensionality of the data would take to long. Given a sample rate of *fs* = 44*,*1*kHz* (usual CD sample-rate) and a length of a song of about *t* = 180*s*, the time domain contains 7938000 data points usually with

16-bit resolution.

*K* = *fs · t* (2.4)

Calculating a FFT with a window size of 1024 samples and a hop size of 512 samples

(resulting in the factor 1.5 in equation 2.5)[1], the full resulting spectrogram would contain 11627 frames with 1024 frequency values per frame. (eq: 2.5)

44100 *samples/s*

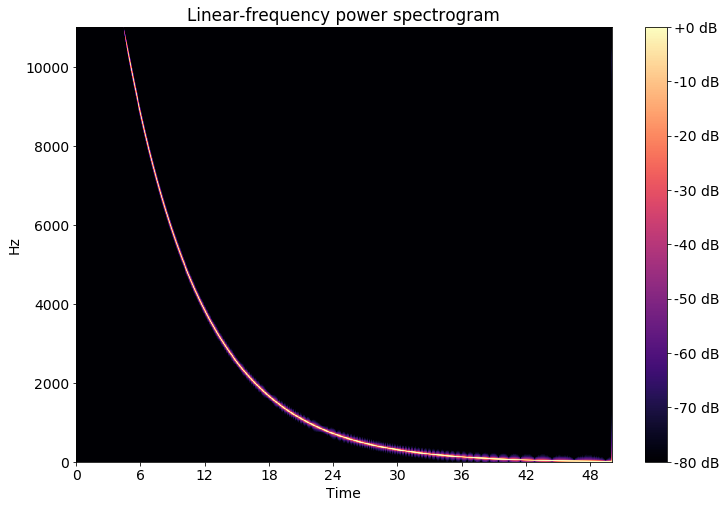
*Nfv* = 1*.*5 *·* ( ) *· t* (2.5)

1024 *samples/frame*

To reduce the dimensionality of the feature vector, a typical approach in MIR would be to calculate the so called Mel Frequency Cepstral Coefficients (MFCCs). They are described in more detail in the next section.

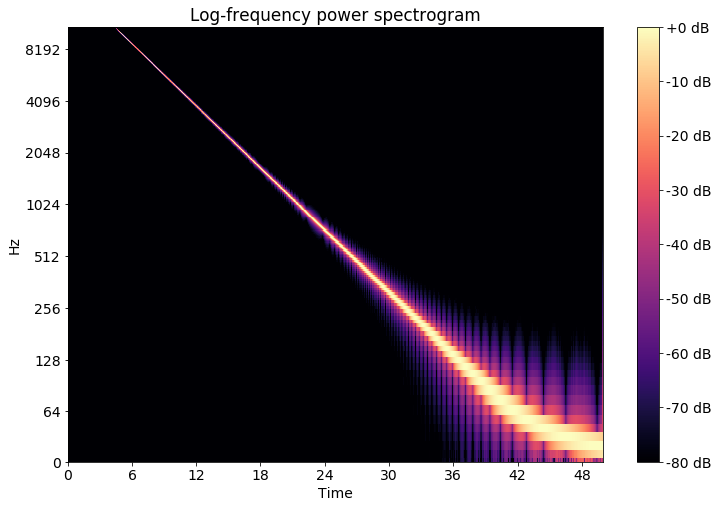
### 2.1.2 MFCC

This section gives a brief overview over the computation of the MFCC as stated in [1, pp. 55ff] because out of all features presented in this chapter the MFCC is the hardest one to grasp because of its abstract nature and hardly visible relatedness to musical features. Figure 2.2 shows the magnitude spectrum of a frequency sweep signal.



(

a) Sweep signal linear



(

b) Sweep signal logarithmic

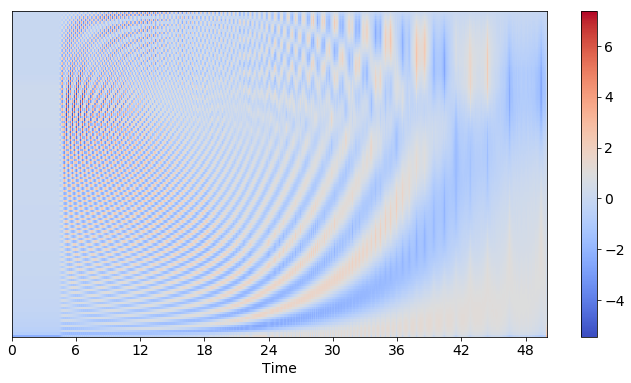
Figure 2.2: Sweep Sinal

First of all the magnitude spectrum is transformed to the Mel-scale by assigning each frequency value to a Mel- band. Doing this a dimension reduction can be done, by assigning multiple frequency values to one of typically 12 to 40 Mel-bands. The resulting vectors are then fed into a discrete cosine transformation (DCT) resulting in the MFCCs for each frame.

*N*X*−*1 *π* 1

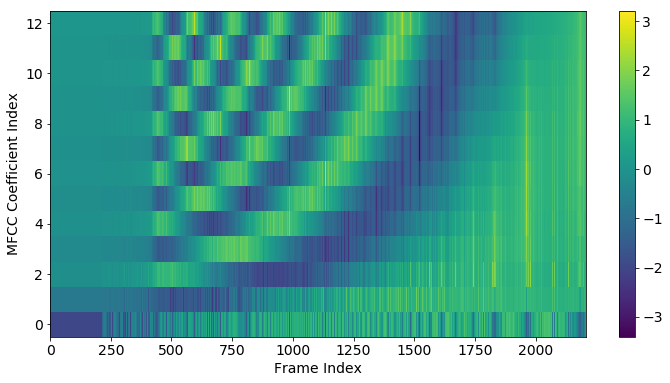
*Xk* = *xncos* (*n* + )*k* (2.6) *n*=0 *N* 2

Figure 2.3a shows the resulting MFCCs with a high resolution of 1024 Mel bands. This is not what in a usual application would be done, because this is nearly as high dimensional as the original spectrogram. Figure 2.3b shows the MFCC reduced to 13 Mel Bands. To better visualize the MFCC features all values are typically scaled to have a standard deviation of 1 and a mean value of 0 per band in the plots.



(

a) MFCC High Resolution

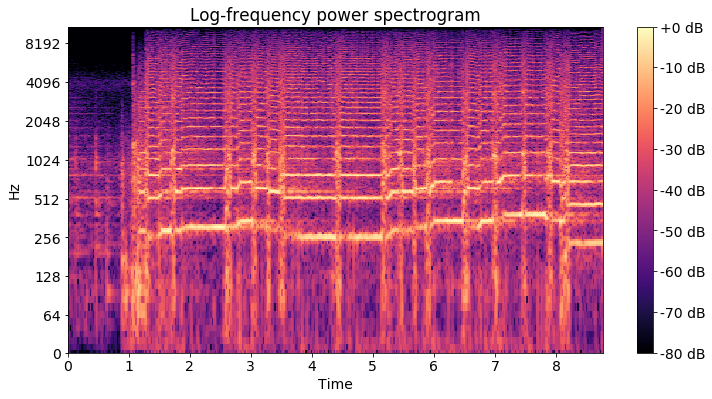


(

b) MFCC 12 Bands scaled

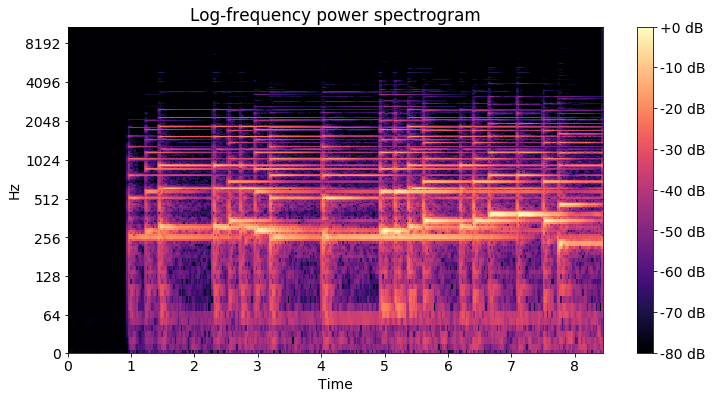
Figure 2.3: MFCCs

MFCCs were found to be suited to represent timbral properties of music [1, p. 55 ff]. To describe a tone, three moments can be used according to [4, pp. 15]: tonal intensity perceived as loudness, the tonal quality perceived as the pitch and the timbre or tonal color as the third moment. Looking at an example melody line played on a electric distorted guitar and a piano, distinct differences can be seen. Due to the physical properties of a string every note played consists of the main frequency (the actually played key) and so called harmonic overtones because of the way a string e.g. in a piano vibrates and the wooden body resonates. Typically the overtones of a piano consist of the main key, the same key a few octaves higher and major thirds and fifths of the octave. Depending on the instrument these harmonics decline faster or slower or don’t appear at all. The electrically amplified guitar amplifies these overtones as well, which is visible in figure 2.4b.



(

a) Guitar

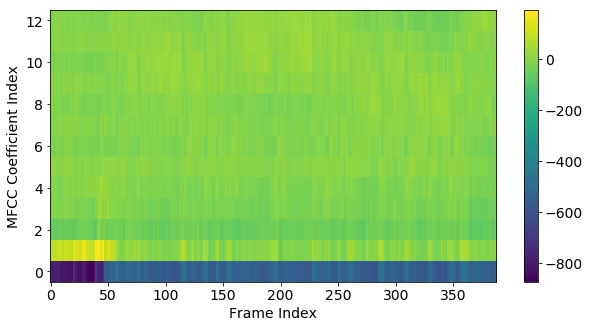


(

b) Piano

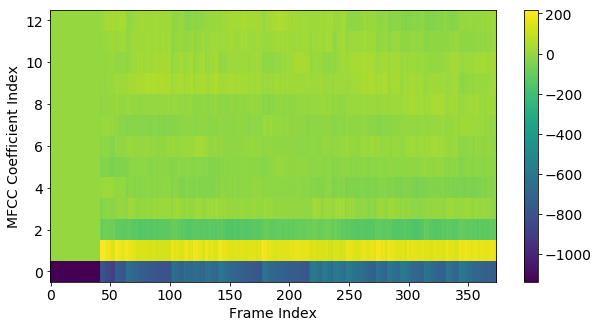
Figure 2.4: Timbre guitar vs. piano

Looking at the MFCCs these differences are also visible. This time the MFCC plots are pictured without the previously mentioned scaling and the mean value and standard deviance of the MFCCs four to 13 are pictured. This calculation of statistical features is later explained in chapter 3.1. Although both times the exact same melody is played in the same tempo, these features vary due to the different timbral properties of the instruments.



(

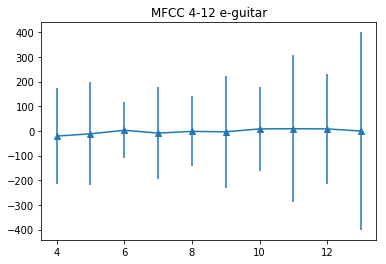
a) Guitar



(

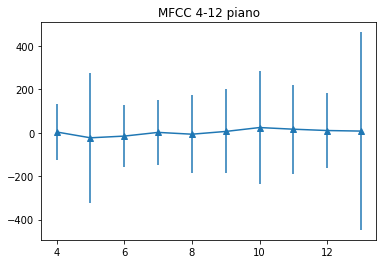
b) Piano

Figure 2.5: MFCC statistics guitar vs. piano



(

a) Guitar



(

b) Piano

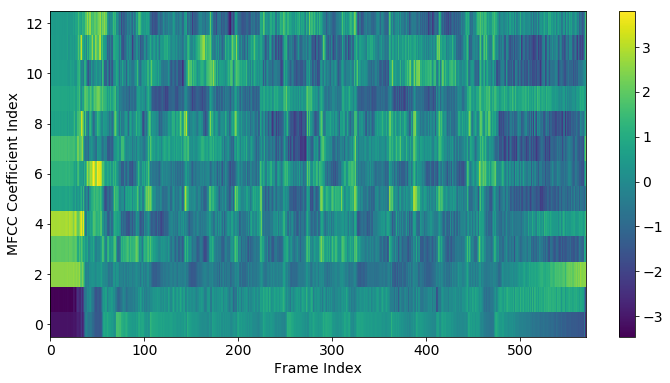
Figure 2.6: MFCC statistics guitar vs. piano

### 2.1.3 other audio features

As another, better comprehensible, higher-level set of features, the chromagram represent the tonal properties of a song. The chroma plot (Figure 2.7b) shows the distribution of the different pitches mapped to the various keys in one octave. The values are normalized to one by the strongest dimension. So if all values are close to one, it is most likely that there is only noise or silence at that frame in the recording, as depicted in the first frames of figure 2.7b. The chromagram has one significant downside, because it is reduced to one octave and thus can not represent the melody of a song to its full extend.

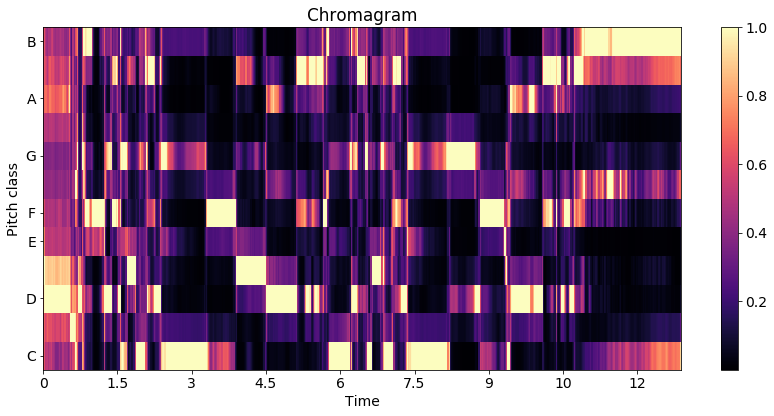
Figure 2.8a figures the pitch curve of the recording. None but the most dominant frequencies are shown. Pitches below a certain threshold are filtered out. In contrast to the chromagram the pitch curve provides information over the whole spectrum and is not limited to one octave. These Pitch curves can be used to estimate and transcribe musical notes from audio data as presented in 2.2.3.

The rhythmic low-level features of a song include the estimation of the overall tempo, beats and onset events. The Plot in figure 2.8b shows the onsets and estimated beats in the first 10 seconds from a recording of the song Layla by Eric Clapton.



(

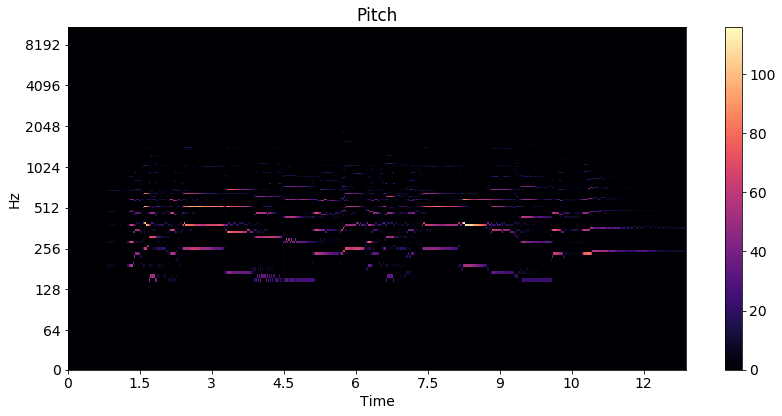
a) MFCC



(

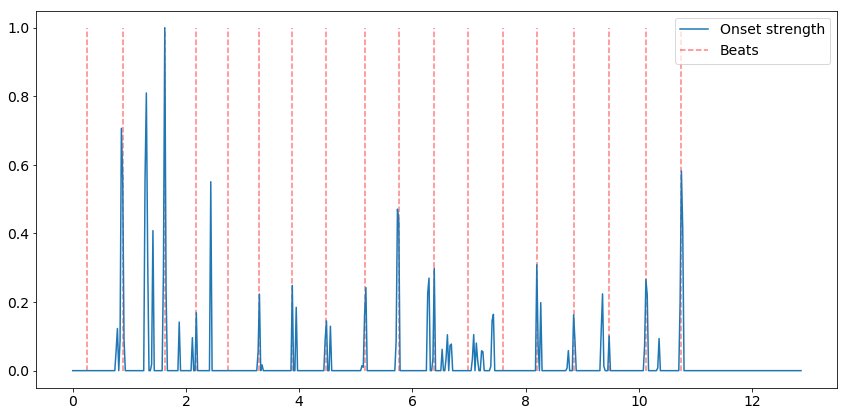
b) Chroma Features

Figure 2.7: Features of the song Layla by Eric Clapton



(

a) Pitch



(

b) Rhythm/ Beat

Figure 2.8: Features of the song Layla by Eric Clapton

## 2.2 MIR Toolkits

### 2.2.1 Low-level audio feature extraction

To extract audio features like the ones presented in section 2.1.1 (mfccs, chromagram, beats, onsets), a wide variety of toolkits is publicly available, a few are presented in [5]. The YAAFE toolkit [6] is able to extract a lot of different audio features like energy, mfcc or loudness directly into the hadoop file format h5 making it ideal for big data frameworks to use. It can be used with C++, Python or Matlab.

The Essentia toolkit [7] is pretty similar to YAAFE, extending it by the calculation of the rhythm decriptors, bpm etc. It can also be used in C++ and Python

The Librosa Toolkit provides similar functionality [8] as Essentia. It is user-friendly, well documented and can be called from a Jupyter-Notebook [9], allowing rapid prototyping and testing of different algorithms. Most of the plots from section 2.1.1 were created using librosa. The code for the extraction of low level features with essentia and librosa is given in chapter 4.1 as well as a performance analysis.

### 2.2.2 Music Similarity

The easiest way to test state of the art music similarity algorithms is to use the open source toolkit Musly [10]. It is based on statistical models of MFCC features and calculates the distances between songs very fast, supporting OpenMP acceleration. It offers the classical mandel-ellis similarity method [11] and a timbre based improved version of the mandel-ellis algorithm using a jenson-shannon-like divergence [12]. More details and a re-implementation of this toolkit is presented in chapter 3.1

As another option, the MIR Toolkit [13] is a toolbox for Matlab [14]. A port to GNU Octave [15] is also available [16]. The short code snippet below is all it takes to compute a similarity matrix based on MFCC features, but the calculation is rather slow.

|  |
| --- |
| mydata = cell(1, numfiles); for k = 1:numfiles  myfilename = sprintf(’%d.wav’, k); mydata{k} = mirmfcc(myfilename); close all force  endfor  simmat = zeros(numfiles,numfiles); for k = 1:numfiles  for l = 1:numfiles  simmat(k, l) = mirgetdata( ... mirdist(mydata{k}, ...  mydata{l}));  endfor  endfor |

Code Snippet 2.1: MIR Toolkit Similarity

### 2.2.3 Melody/ pitch extraction

To test the various pitch extraction toolkits, a piece by Rachmaninoff and by Beethoven was used. The first three bars of Rachmaninoffs Prelude can be found in figure 2.9a

Figure 2.9b shows the first five bars of Beethovens Bagatelle in A Minor (Fur Elise).¨

The first toolkit tested is Aubio [19]. The result can be seen in figure 2.10a and figure



(

a) Rachmaninoff Prelude in C Sharp Minor

[17]



(

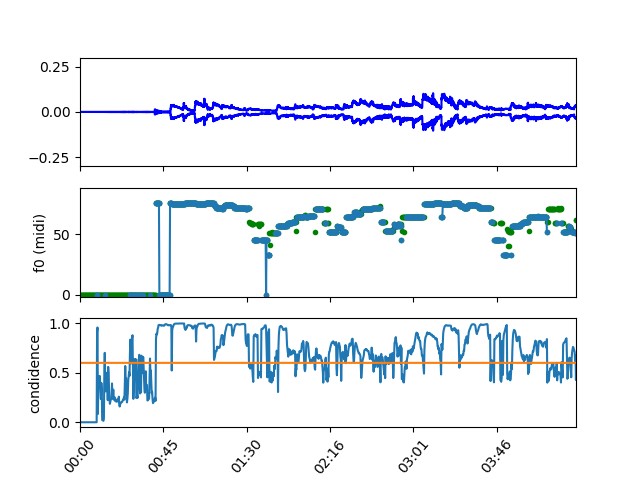
b) F

¨

ur Elise [18]

Figure 2.9: Original Scores

2.10b The upper subplot shows the waveform of the first few seconds of each piece. The second plot figures the estimated pitch with green dots. If the pitch is zero, then no pitch could be estimated, most likely because the associated frame contains silence. The blue dots resemble the estimated pitches, where the confidence (shown as the blue graph in the third subplot) is above a certain threshold (the orange line).

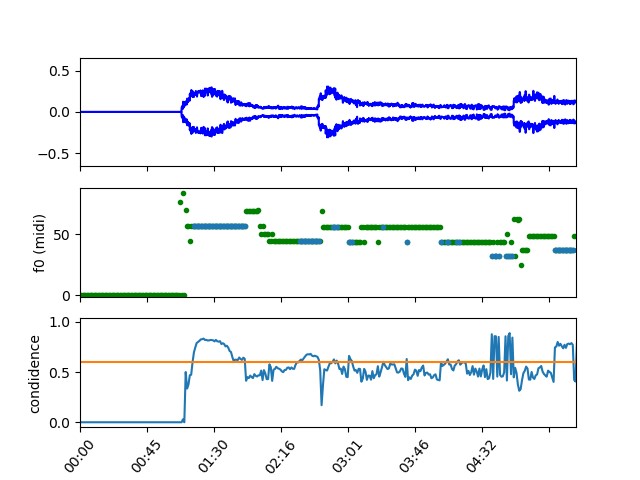


(

a) F

¨

ur Elise Aubio Pitch



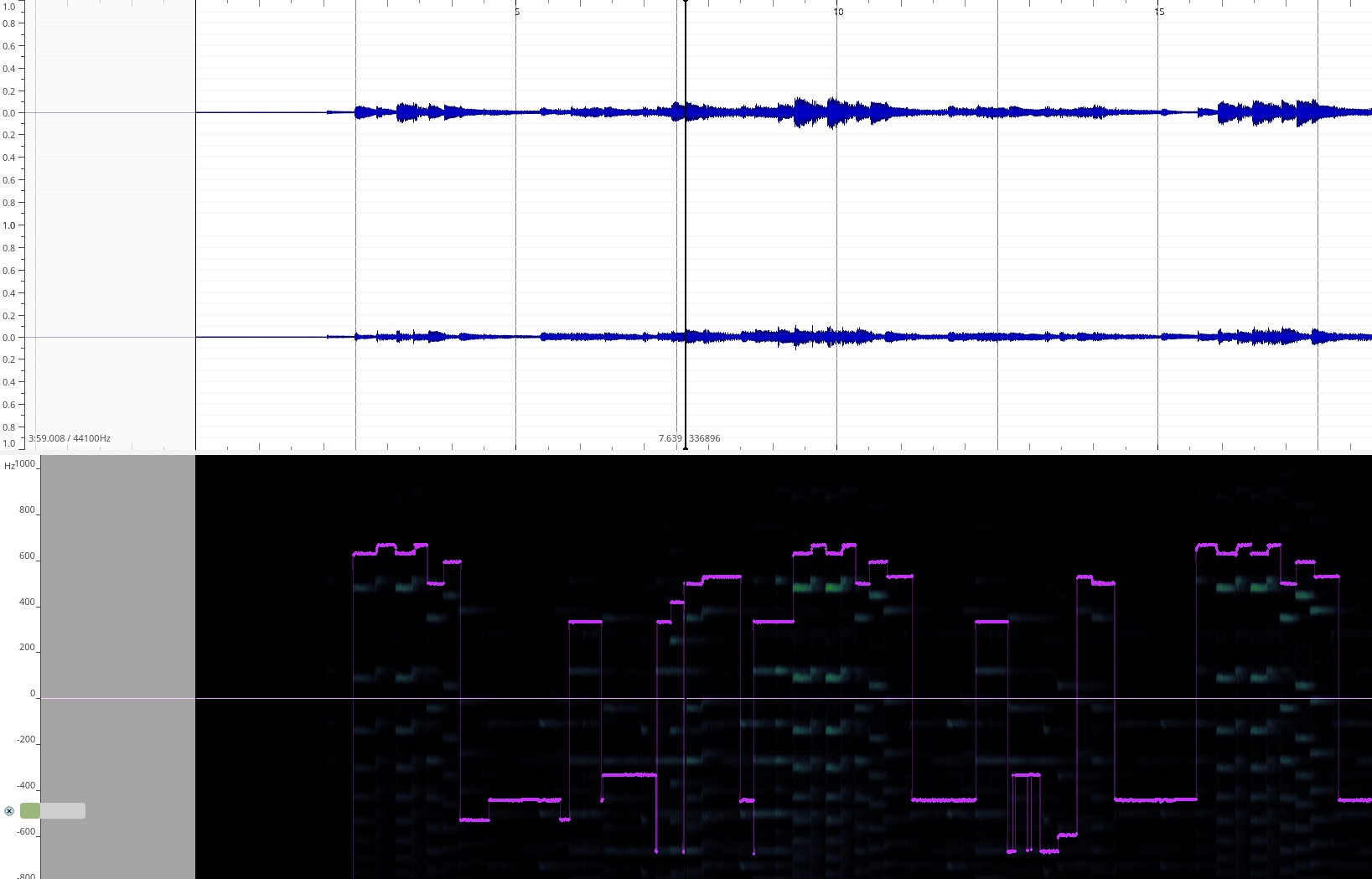
(

b) Rachmaninoff Prelude Aubio Pitch

Figure 2.10: Aubio

The other melody extraction tool is Melodia[20], which is available as a VAMP plugin and can be used together with the Sonic Visualizer[21]

The results are shown in figure 2.11a and 2.11b. The purple line is the estimated pitch, however there are large jumps between different octaves of the harmonics.

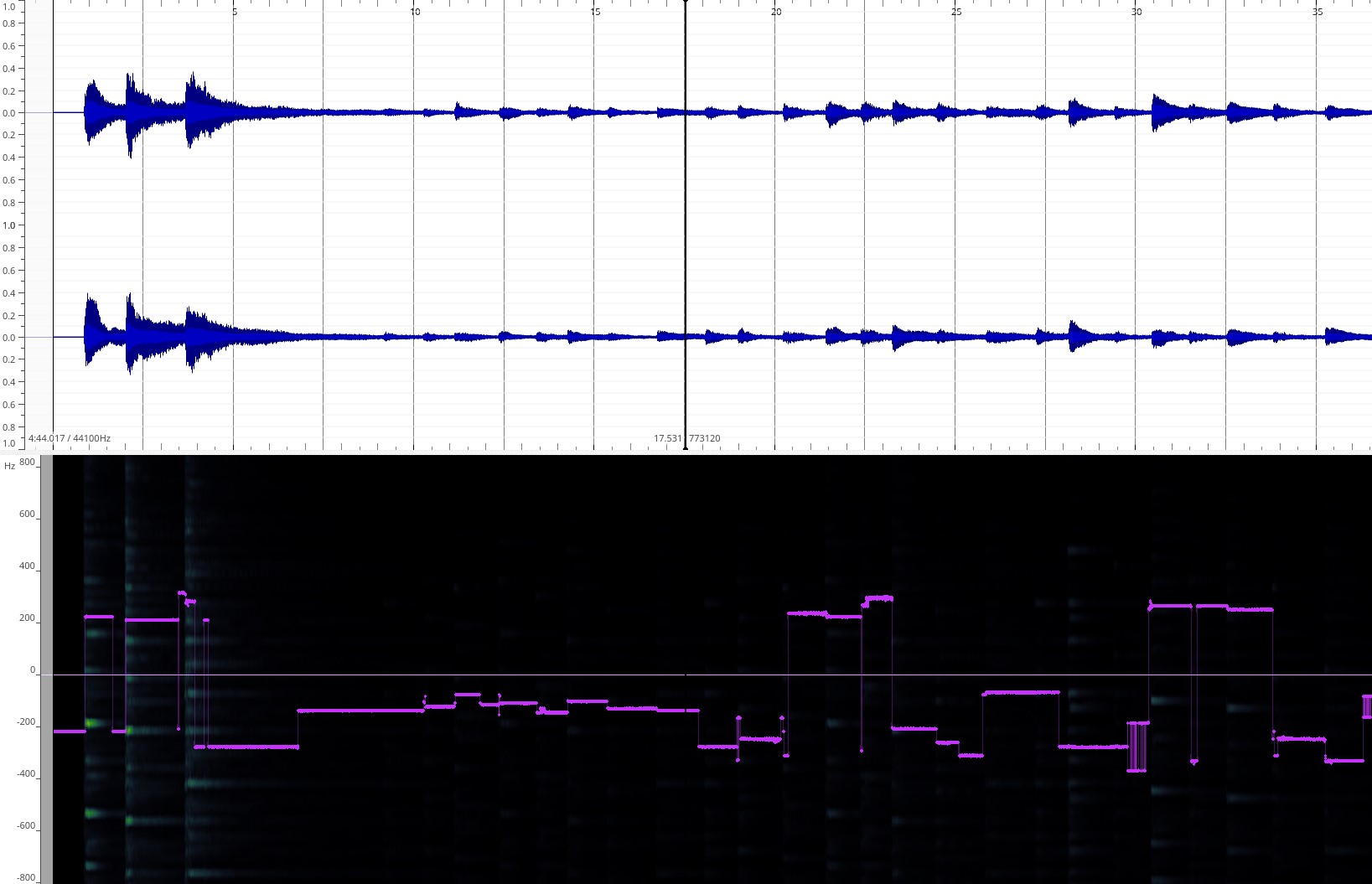


(

a) F

¨

ur Elise Melodia Pitch



(

b) Rachmaninoff Melodia Pitch

Figure 2.11: Melodia

Sadly the conversion to MIDI does not work flawlessly. It is clearly visible in figure 2.12 that the transcription does not work accurately enough even for a classical music piece with only one instrument. Figure 2.12a shows the output of a python script using the Melodia VAMP plugin to calculate a MIDI file containing the main melody line and figure 2.12b shows the transcribed MIDI notes from Aubio. The detected melody lines are jumping between different octaves and finding the right threshold for the separation between silence and detected notes turns out to be problematic as well.



(

a) F

¨

ur Elise Melodia MIDI



(

b) F

¨

ur Elise Aubio MIDI

Figure 2.12: Transcription

## 2.3 Music Similarity Measurements

### 2.3.1 Timbre based

As done by: [22]

The proposed approach by [22] is, to take mfccs as low level features and compute statistical features like mean, standard deviation and covariance of the different mfccs to reduce dimensionality before computing similarities. Various similarity measurements are proposed and evaluated in chapter 3.1 The algorithm used in this thesis is described in [22, pp. 17ff]. The framework used to determine the song similarity for the first tests in chapter 3.1 is the MUSLY toolkit [10].

### 2.3.2 Pitch based

As done by: [23]

The proposed approach by [23] is, to take mid-level features like the chromagram or estimated main pitchline instead of high-level features like sheet music or low-level features like gaussian mixture models of MFCCs. A more detailed analysis of this topic is given in chapter 3.2

### 2.3.3 Rhythm based

As done by: [24]

Rhythm based music similarity algorithms use timing information of various events as a baseline. For example low-level features like the onset and beat data from the plot in Figure 2.8b could be used as a starting point for rhythmic similarity retrieval. More advanced approaches make use of the beat histogram, rhythm histogram and rhythm patterns. An in depth overview is given in chapter 3.3

### 2.3.4 Metadata based/ Collaborative Filtering

As done by: [25] and [26] using the Million Song Dataset (MSD) [27]

In 2012 the MSD Challenge was brought to the MIR community. The researches were challenged to give a list of song recommendations based on a large set of user data. So if user X listened a lot to artist A and artist B and user Y listens mostly to artist A and artist C, then maybe user X would like artist C as well. These kind of collective listening behavior based recommendations are called collaborative filtering [1, p. 192f.] and are pretty common in large music streaming services, although not necessarily representing direct musical similarity. These kind of recommendation systems tend to propose commonly well known artists rather than not so well known ones biasing the result. On the other hand these kind of similarity algorithms can work very fast and efficient in a Big Data environment. The usage of annotations and metadata information like genre and artist based recommendations are common as well. The recommendation of songs based on the lyrics and also hybrid recommendation systems that combine lyrics, metadata and collaborative filtering are possible. However all of these recommendation strategies are not directly based on musical features and are not considered for this thesis. They are however a possible addition for a hybrid recommendation engine for future research.

### 2.3.5 Note based

As done by: [28]

For comparing musical pieces by their symbolic represenation (notes, tabulatures etc.) different text retrieval methods could be used. The MIDI datatype could be used, as it is a form of digital representation of music information. [28] uses a variation of the Levinshtein distance measurement. The problem with notation based algorithms is, that there are not many datasets available containing audio and MIDI information. As shown in 2.2.3 the automatic transcription of notes from raw audio does not work flawlessly. There is ongoing research to automatically annotate musical notes with the help of neural networks.[29] In chapter 3.3 an attempt by Xia (et. al) [30] to extract note information as text features from chromagrams and calculating the similarity by using the levinshtein distance is shown and evaluated.

### 2.3.6 Genre specific features

As done by: [31] for indian art music, by using 560 different combinations of different features. They state that: ”We evaluate all possible combinations of the choices made at each step of the melodic similarity computation discussed in Section 2. We consider 5 different sampling rates of the melody representation, 8 different normalization scenarios, 2 possibilities of uniform time-scaling and 7 variants of the distance measures. In total, we evaluate 560 different variants” [31, p. 3]. This evaluation showed, that the choice of features and parameters for music similarity measurement is a critical point.

In Rock, Pop and Metal music, extraction of different guitar playstyles would be imaginable. Guitar Tab Extraction [32] Toolkits could be used to extract information if the guitar in a song is mostly plucked or strummed for instance. Or if there are Hammer-on/ Pull-off/ side bending or tapping techniques used. In classical music, the play style of the string section of an orchestra could be taken into consideration.

### 2.3.7 summary

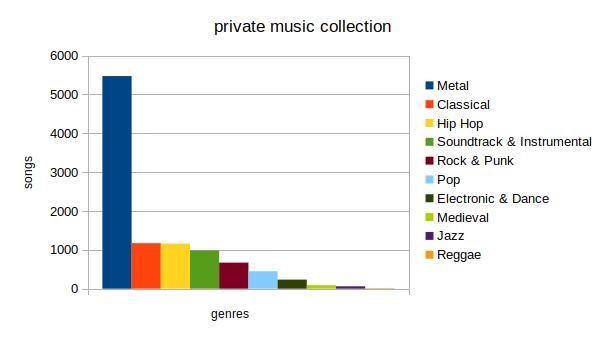
In this thesis music similarity measurements based on three different types of features are evaluated. The first is based on MFCCs to represent timbral features of the songs and therefor offering a set of features to make recommendations that are similar in tone color and should be able to make recommendations inside the boundaries of different genres. The second is based on chroma features/ note information collapsed to one octave to represent a measurement of melodic similarity. With these features the detection of cover versions should be possible. The third set of features is based on the rhythmic properties of a song. This should enable recommendations of songs with the same tempo and rhythmic structure, possibly also enabling the recommendation of songs within the same genre.

The usage of MIDI-files is not considered further due to the rather poor performance of score extraction tools for songs with multiple instruments and melody lines. Also the melodic component of the songs is already represented by the chroma features, although that limits the representation to one or at most a few octaves. Collaborative filtering is left out due to the fact that it does not necessarily represent the musical features and properties but the personal taste of music. And secondly no dataset with the required information and music files was found (2.4). Genre specific features are also not an option because this field is not yet very well researched and the development of algorithms and the extraction of features would go beyond the scope of this thesis.

## 2.4 Data aggregation

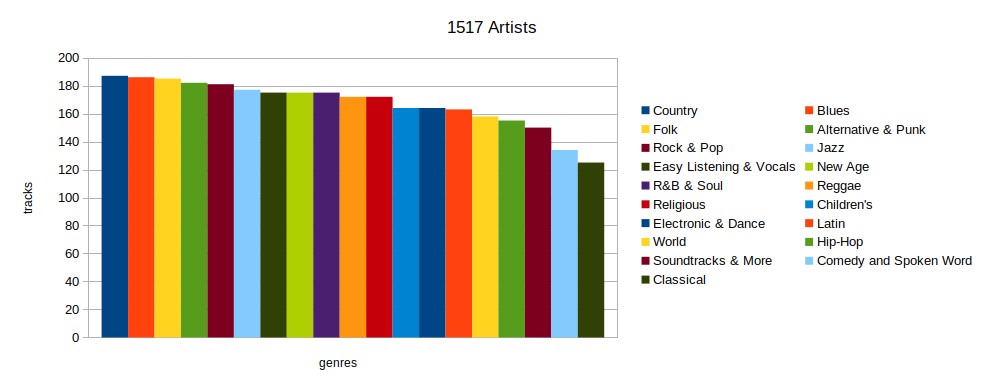
To evaluate the music similarity algorithms and metrics a lot of music data is needed.

### 2.4.1 Datasets



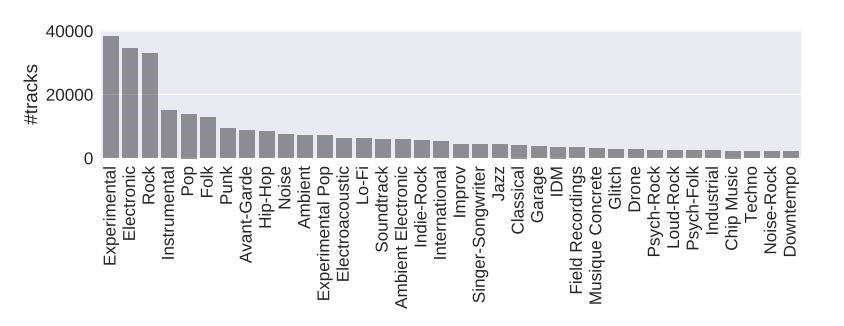
(

a) private music collection



(

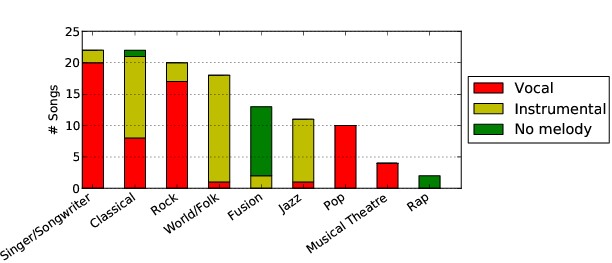
b)1517artists



(

c) fma [33, p.

4]



(

d) MedleyDB [34, p.

2]

Figure 2.13: Datasets

##### Free Music Archive

The largest dataset is the Free Music Archive- dataset (fma) consisting of 106733 different songs totalling an amount of nearly one terabyte of music data from all kinds of different music genres [33]. There is also a lot of metadata information available for most of the songs.

##### Private music collection

The private music collection used in this work consist mainly of metal music. The music was legally purchased, all rights belong to the respective owners. The distribution of different songs per genre for this dataset is visualized in figure 2.13a.

Additionally a private recording dataset was used, consisting of ambient recordings and self produced music. Most of this music can be downloaded from soundcloud [35].

The private music collection is fully cataloged and the according pdf file is the appendices.

##### 1517 artists and Musicnet

Another source of music is the Musicnet dataset [36]. It includes 330 pieces of classical music with musical notes as annotations. Other sources of musical information would be the 1517-Artists dataset containing 3180 songs of multiple genres. [37]

##### covers80

For a cover song detection system, the covers80 dataset is available [38] containing

80 original songs mostly from the musical genres rock and pop and 84 cover versions. These cover versions tend to differ significantly in musical style, rhythm and timbre from the original.

The ability to detect cover songs or different versions/ recording of a musical piece could be a good measurement for the efficiency of music similarity algorithms. In the next section one test case is presented, showing, that a MFCC based music similarity algorithm isn’t able to detect different recordings of the same piano piece as most similar to each other.

##### MedleyDB

For a melody/ pitch based similarity analysis, multitrack datasets could provide useful data, due to the fact that the pitch estimation can be done instrument by instrument. Datasets available are the MedleyDB [34] and MedleyDB2 [39] datasets as well as the Open Multitrack Dataset [40] currently consisting of 593 multi-tracks in which the MedleyDB dataset is already included, leaving 481 other tracks for analysis.

##### Overview and other sources

The music sources and amounts of songs used for the task at hand is listed in table 2.1.

|  |  |
| --- | --- |
| fma | 106.733 Songs |
| private | 8484 Songs |
| 1517 artists | 3180 Songs |
| Maestro | 1184 Songs (piano) + MIDI |
| musicnet | 330 Songs (classical) + note annotation |
| Open Multitrack Testbed | 593(481) Songs/ Multitracks |
| covers80 | 164 Songs (80 originals + 84 covers) |
| MedleyDB | 122 Songs/ Multitracks |
| MedleyDB2 | 74 Songs/ Multitracks |

Table 2.1: music datasets

### 