MEE05:30

Music Information Retrieval

A case study of MIR in modern rock and metal music

Szymon Janusz Szczepański
Michał Stanisław Szyca

This thesis is presented as part of Degree of

Master of Science in Electrical Engineering

Blekinge Institute of Technology
October 2014

Blekinge Institute of Technology

School of Engineering

Department of Applied Signal Processing

Supervisor: Mr. Magnus Berggren , Dr. Sven Johansson

Examiner: Dr. Sven Johansson

ABSTRACT

"Music Information Retrieval (MIR) is a multidisciplinary research endeavor that strives to develop innovative content-based searching themes, novel interfaces, and evolving networked delivery mechanisms in an effort to make worlds vast store of music accessible to all" (Downie, summer 2004). The purpose of this study is analysis of some MIR algorithms and their efficiency on modern rock and metal songs. To do so, one algorithm for four of most commonly MIR uses was chosen, described and evaluated on full songs or fragments of them. Music separation part proved that it is possible to effectively separate repeating background. Audio fingerprinting experiments show that higher quality of records generate significantly better results. Modern metal music turned out to be difficult for tempo estimation and chord recognition was working. All results show that modern rock and metal music, even though complicated, can be analyzed but still needs more research in this field to make it effectively.

ACKNOWLEDGMENTS

We would like to express our gratitude to both Mr Magnus Berggren and Doctor Sven Johansson for taking supervisory over our thesis.

TABLE OF CONTENT

1. INTRODUCTION

Background and Problem Statement

As the Music Information Retrieval field is growing, a lot of algorithms, solutions, articles and other papers are created. Most of this work is evaluated on databases consisting mixed genre of music. In these databases rock and metal music is considered usually as one or two genres without separating it to subgenres. It is hard to find a work considering only rock and metal music or its subgenres (Tsatsishvili, November 2011) (Mulder, July 2014).

The purpose of this thesis is evaluation of some Music information Retrieval solutions on modern rock and metal music and its few subgenres to check if algorithms designed for common use can be effectively used on such music.

Division of Work

Szymon Janusz Szczepański prepared parts:

- Track Separation
- Audio Fingerprinting

Michał Stanisław Szyca prepared parts:

- Tempo Estimation
- Chords Recognition

All other parts were prepared together.

Music Information Retrieval

"Music Information Retrieval (MIR) is a multidisciplinary research endeavor that strives to develop innovative content-based searching themes, novel interfaces, and evolving networked delivery mechanisms in an effort to make worlds vast store of music accessible to all" (Downie, summer 2004). In other words Music Information Retrieval tries to make music more accessible to listeners, easier to create to musicians and more analyzable to professionals.

The term "Music Information Retrieval" was probably first time used by Michael Kassler in book (Kassler, spring-summer 1966). In his part about MIR he described it as programming language used to extract specific data from musical data. From 1966 this field was slowly developing with

boom starting around year 2000. During these years researchers have developed many methods for analyzing audio and obtaining many information from them. It is possible to transcript, read and play notes, recognize sounds, chords, look for similarities between audio objects, analyze broadcasts, estimate tempo and many others. Achieving a lot, MIR has reached point where most of metadata- and content-based information can be retrieved from audio.

Nowadays Music Information Retrieval benefits from disciplines like biology and psychology. It is no more pure computer science field. Researchers focus on psychology, musical taste, emotions and subjective musical taste of listener now. That is next step in evolving Music Information Retrieval (Wiering, December 2006). Wiering even suggested changing a term to Music Retrieval, because information part is useful mostly for professionals, other humans benefit more from other parts. Interesting talk between psychologist and MIR researcher on this topic was presented on ISMIR 2012 (International Society for Music Information Retrieval Conference) (Aucouturier, et al., 2012). There are methods that try to benefit from those approaches like social tags (Lamere, November 2008). Social tags are subjective, undefined, short phrases describing songs, that collected in one database can be used as big information centre.

Typical non-professional listeners cause problems. They would like to listen to music they like but it is hard for people without musical knowledge to define what music they would like to listen. Next problem is how to present results of MIR to listener. Raw information like chords, tempo and so on are unintuitive. There are some studies on finding intuitive output methods (Goto, et al., September 2005) (Pampalk, et al., August 2003) (Vignoli, et al., August 2004).

Track Separation

Track/source separation in Music Information Retrieval describes algorithms and solutions for extraction of certain instrument or sound from polyphonic audio file. In our thesis we decided to analyze track separation algorithm basing on repetition in music. This algorithm separates audio file into repeating background and non-repeating foreground.

Repetition is basis of music. Sounds repeated over time create a background which makes singing and playing instruments easier and songs more enjoyable for listeners. People usually nod a head or stamp a feet to the background of a song intuitively, without thinking about it, because it is easier to repeat same thing than follow something changing over time. In music genres like pop, most songs are created by overlaying repeated accompaniment by vocals.

Repetition is used in MIR (Music Information Retrieval) in many ways. Songs segmentation and rhythm estimation can be done with a help of repetition analysis. Repetition can be also used for track separation. It is not common way of doing it but Zafar Rafii and Bryan Pardo (Rafii, et al., January 2013) proved that there is a simple way for beat separation using repetition property. This method and its efficiency is described in Track Separation chapter 2.

Effective and fast method for music and vocal separation in song will find many applications not only in MIR but also in real life. Starting from karaoke singing, through teaching music and ending with audio post processing for already mixed songs.

Audio Fingerprinting

The Term "Audio Fingerprinting" comes from human fingerprint method for identifying people. The same as in human case, audio fingerprints obtain less information than its origin but this information is so unique it allows to identify origin. Using fingerprints reduces required space for storage, increases comparison efficiency and speeds up searching though database. Usually fingerprint is connected to meta-data only, all irrelevant information is omitted. Fingerprint systems consist of method for extracting fingerprints and method for identifying them (Haitsma, et al., August 2002).

Audio fingerprinting have many applications. One of them is broadcast monitoring (Jang, et al., June 2006) (Neves, et al., March 2009). Using correct methods it is possible to create playlists from live radio broadcasts and analyze them. This way artist, sponsors or advertisers can monitor if radio station fulfill the conditions of contract. Next usage of audio fingerprinting algorithms is track identifying, first thing that comes to mind when you ask somebody about audio fingerprinting. Analyzing part of the song to find its artist, title, publisher, date of release and so on. Automatic music library organization can also benefit from audio fingerprinting. Big digital audio collections can be automatically organized ignoring wrong tags and meta-data. Last but not least is a possibility of controlling files people upload to free servers, p2p networks or clouds (Shrestha, et al., August 2004). This way changing the name of a track will not work because all track would be analyzed by their content.

The purpose of this part is to check how effective is one of common audio fingerprinting algorithms, Shazam algorithm (Wang, January 2006), when working on modern rock and metal songs.

Tempo Estimation

Tempo estimation is one of the fundamental processes in music information retrieval. In the project we use the bank of comb filters approach to achieve it.

There are several ways to detect tempo in music. The main difference about them is that they use the autocorrelation or the bank of comb filters approach to achieve that goal. Approach used in the project is the bank of comb filters (Scheirer, January 1998).

The bank of comb filters and autocorrelation are both good approaches to check the periodic changes in the signal in this case signal tempo. There are two advantages of using the comb filter method over the autocorrelation. (Scheirer, January 1998). The comb filter method not only give us ability to determinate tempo it is also phase preserving thank to what we can also determinate the

phase of the signal that we are examining. This feature of the approach can be useful in making a higher level processing algorithm.

Another disadvantage of the autocorrelation is that it do not have another ability of comb filtering that is “encoding the aspects of rhythmic hierarchy” (Scheirer, January 1998). It means that a comb filter when it is tuned in a tempo τ it has a response at tempo τ and also some smaller responses ant other tempos that are multiples of τ tempo (2 τ or 3 τ), fractions (τ/2, τ/3) or (3/2 τ, 2/3 τ etc.), the autocorrelation has those only for fractions of the tempo. For example an autocorrelation based algorithm do not get any sense of tempo at 240 beats per minute (bpm) while examining a signal whit 120 bpm tempo. The comb filter give as that sense but reduce it when compare whit 240 bpm signal.

The advantage of using the autocorrelation is that it has better memory usage than comb filters. It is because during the examination of the signal every comb filter needs to have its own delay line. Which effects with a bigger memory usage but also gives the energy at each phase angle of each delay. Autocorrelation does the process on one delay line, due to that it does not need to use so much memory (that fact also makes autocorrelation zero phase). Unlike the comb filters it give an output that is summary of each phase energy.

Chords Recognition

One of the problems in music information retrieval is chord recognition. One of the best ways to deal with it, is to use the neural network. The neural network method feature the pitch class profile vector. The vector provides only a 12 element information on the semi-tone values, but it is adequate for chord recognition task.

Of course there are other methods to perform this task, well most of them is based on pitch class profiling to change the chord to a form suitable to recognize but the process of recognition is based on very complex and memory using methods.

Thesis Outline

Chapter 1 describes basis and origin of Music Information Retrieval and some of its fields. In chapters 2, 3, 4 and 5 track separation, audio fingerprinting, tempo estimation and chord recognition algorithms were described accordingly. In these chapters several experiments were also conducted to evaluate effectiveness of these algorithms on modern rock and metal music. Chapter 6 is summary and conclusion.

2. TRACK SEPARATION

REPET method

REPET (Repeating Pattern Extraction Technique) is a method for analysis of music files basing on repetition in music (Rafii, et al., January 2013). It can be divided into three steps: identification of the repeating period, modeling and extraction of repeating pattern. The three steps we described below.

A. Identification of repeating period

First step in REPET technique is obtaining spectrogram X of music file. It is achieved by calculating spectrogram STFT (Short Time Fourier Transform) of signal x and plotting it time-frequency plane (Kozek, et al., July 2007). DFT is usually enough for analysis of signal but not always. Single Fourier transform does not allow to observe changes of spectral content. It represents spectrum for whole audio file. Short Time Fourier Transform presents variation of single Fourier transform over time.

Next step is discarding symmetric part and keeping the DC component of spectrogram X. After that magnitude spectrogram V is obtained by taking absolute value of modified spectrogram X.

To emphasize peaks in magnitude spectrogram V, power spectrogram V^2 is calculated by element-wise square of V. For stereo audio files power spectrogram V^2 is averaged over both channels.

Then autocorrelation function calculated over rows of spectrogram V^2 is used to create matrix B. Autocorrelation is a function of similarity between signal and its lagged version. Beat Spectrum vector *b* is obtained by calculating mean over the rows of B, normalizing it by its first element (no lag) and discarding first term representing similarity with not lagged signal. The effect is vector *b* presenting peaks periodically repeating over time.

$$B(i,j) = \frac{1}{m-j+1} \sum_{k=1}^{m-j+1} V(i,k)^2 \, V(i,k+j-1)^2$$

$$b(j) = \frac{1}{n} \sum_{i=1}^{n} B(i,j)$$

$$b(j) = \frac{b(j)}{b(1)}$$

$$\textbf{for } i = 1 \ldots I \text{ (frequency) } \textbf{where } I = \frac{N}{2} + 1$$
$$\textbf{for } j = 1 \ldots J \text{ (lag) } \textbf{where } J = \#time\ frames$$

Finally repeating period *p* can be obtained from beat spectrum vector *b*. Longest lags in autocorrelation function are uncertain and we are looking for periods that repeat at least 3 times over reliable values so period *p* is looked for over first 1/4 length only. To find it we are looking for period that has the highest mean accumulated energy over its integer multiples. Exact algorithm is presented below.

$l \leftarrow$ length of b after discarding the longest $\frac{1}{4}$ of lags
$\delta \leftarrow$ fixed deviation for possible shifted peaks
$J \leftarrow$ empty array of length $\left\lfloor \frac{l}{3} \right\rfloor$

for each possible period j in the first $\frac{1}{3}$ of b ***do***
$\Delta \leftarrow \left\lfloor \frac{3j}{4} \right\rfloor, \qquad I \leftarrow 0$

for each possible integer multiple i of j in b **do**
$h \leftarrow argmax\ b(k), \qquad$ **where** $k\epsilon[i-\delta, i+\Delta]$

if $h = argmax\ b(k), \qquad$ **where** $k\epsilon[i-\Delta, i+\Delta]$ **then**
$I \leftarrow I + b(h) - mean\ b(k),$

end if

end for

$J(j) \leftarrow \frac{I}{\left\lfloor \frac{l}{j} \right\rfloor}$

end for
$p \leftarrow argmax_j J(j)$

Algorithm 1 Finding repeating period *p* from beat spectrum *b*

B. Modeling repeating pattern

Having repeating period *p* calculated it is possible now to divide spectrogram *V* into *r* segments of *p* length. Now a method to obtain model of repeating background pattern is required. Segment model *S* is element-wise median over those segments.

$$S(i,l) = median\{V(i, l + (k-1)p\} \textbf{ where } k = 1 \dots K$$
$$\textbf{for } i = 1 \dots I(\text{frequency}) \text{and } l = 1 \dots L \text{ (time)}$$
$$\textbf{where } p = period\ length \text{ and } K = \#segments$$

C. Extraction of repeating pattern

The last part is extracting a repeating pattern. Repeating spectrogram model *W* is derived by taking element-wise minimum of *S* and *r* segments of *V*.

$$w(i, l + (k-1)p = min\{S(i,l), V(i, l + (k-1)p)\}$$
$$\textbf{for } i = 1 \ldots I, l = 1 \ldots L, \text{and } k = 1 \ldots K$$

Next step is creating time-frequency mask M. Soft mask was chosen as default because it was giving better results for most experiments. Soft time-frequency mask *M* is get by element-wise normalizing *W* by *V*, so mask is going to have values close to 1 for cells that represent repeating background and close to 0 for cells representing foreground.

$$M(i,j) = \frac{W(i,j)}{V(i,j)} \textbf{ with } M(i,j) \in [0,1]$$
$$\textbf{for } i = 1 \ldots I \text{ (frequency) and } j = 1 \ldots J \text{ (time)}$$

Last step is symmetrizing time-frequency mask *M*, multiplying it by Short Time Fourier Transform *X* and applying ISTFT (Inverted Short Time Fourier Transform) to it. The final result is background calculated for mixture *x*. To obtain foreground simply abstract background from mixture *x*.

Original REPET algorithm

Experiment settings and results

To check overall performance of algorithm on rock and metal songs 5 parts of songs from different music genres were chosen:

- Antigama - Collapse
- Blood Red Throne - In Hell I Roam
- Default - Sick & Tired
- Devourment - F***ed With Rats
- Kvelertak - Apenbaring

Parts were chosen to have repeated background over time. The experiment was conducted for 5 different length of hamming window (10ms, 40ms, 100ms, 500ms and 1s), all with 50% window overlap. Music signals are considered stable after 40ms windowing and other lengths were chosen to check if changing window length may improve results for metal and rock music. Separation was done by using original REPET code in m-file dated September 2013 from project website. Audio files are parts of songs chosen for no changing background over time.

It is hard to analyze and compare separation effects precisely by ear. To make it easier Emiya *et al.* presented PEASS Toolkit for Matlab for evaluation of Audio Separation (Emiya, et al., February 2011) (Vincent, March 2012). It consists of commonly used BSS Eval Toolbox and Perceptual Scores considered more similar to human analysis. **SDR (Source-to-Distortion Ratio)** and **OPS (Overall Perceptual Score)** are most important indicators of separation in those two methods. Higher values are better. Results are presented in Table 1 and Table 2. Columns present results for different artists and are divided for beat and vocal results. Rows divide results for different length of windows.

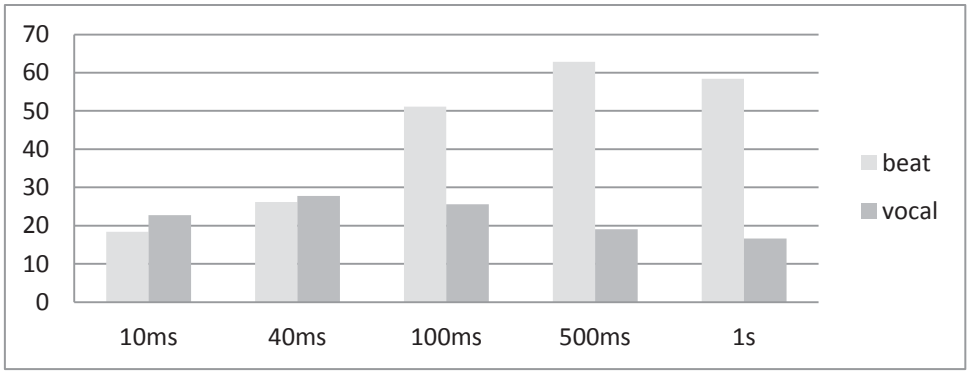
OPS	**Antigama**		**Blood Red Throne**		**Default**		**Devourment**		**Kvelertak**	
	beat	vocal	beat	vocal	beat	vocal	beat	vocal	beat	vocal
10ms	18,7055	20,0992	17,6123	20,4879	17,3188	**30,323**	18,9669	21,034	19,2621	21,6414
40ms	21,5374	**25,147**	26,0409	**30,1469**	33,7149	25,7014	25,9405	**26,297**	23,8047	31,3579
100ms	63,8084	20,2738	49,9202	26,3925	40,26	28,2344	62,8138	19,8195	39,0133	**33,2589**
500ms	**72,7731**	21,2976	**58,147**	17,9348	**55,3827**	21,3056	**66,4079**	16,1682	**61,3461**	18,7312
1s	58,9734	16,1407	56,3581	16,228	52,5415	18,5872	63,3002	14,9049	60,86	17,2184

Table 1 OPS results for original REPET

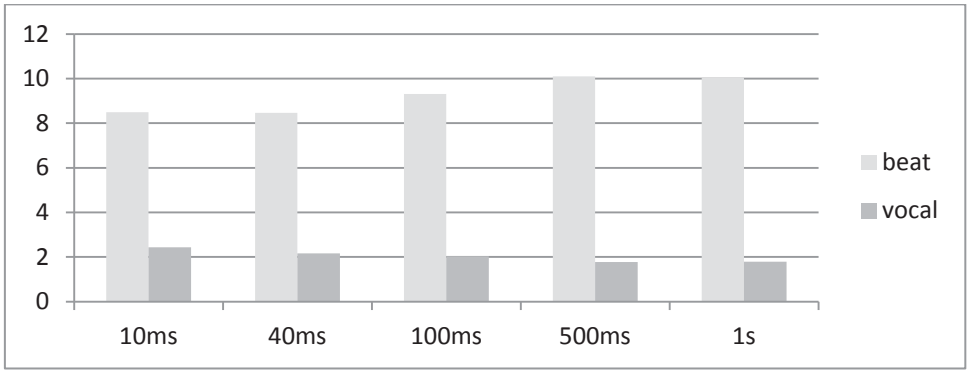
SDR	**Antigama**		**Blood Red Throne**		**Default**		**Devourment**		**Kvelertak**	
	beat	**vocal**	**beat**	**vocal**	**beat**	**vocal**	**beat**	**vocal**	**beat**	**vocal**
10ms	9,6191	**1,9479**	8,8667	**2,1492**	6,687	**3,1412**	8,5876	**2,4916**	8,6935	2,4347
40ms	9,3831	1,8913	8,455	1,9876	8,5155	2,1033	8,281	2,2214	7,6746	2,5985
100ms	**11,4174**	1,3896	10,1502	1,608	7,6779	2,5247	10,373	1,6156	6,9489	**3,0143**
500ms	11,0866	1,5198	**10,3912**	1,629	8,7492	2,2197	10,5469	1,6257	9,6927	1,9199
1s	10,721	1,5662	10,166	1,7065	**9,0255**	2,1546	**10,6354**	1,6146	**9,6938**	1,9265

Table 2 SDR results for original REPET

To make results more clear to read, average values for each window length were calculated and presented below on Graph 1 and Graph 2.



Graph 1 OPS Results



Graph 2 SDR Results

Conclusion

Analyzing received results following conclusions were deduced:

- Repeating background is better separated than non-repeating foreground using REPET method.
- Repeating background is better separated with REPET method using longer window lengths. Best results for background were obtained using 500ms window.
- Non-repeating foreground is better separated with REPET method using very short window lengths.

Adaptive REPET

Original and unmodified REPET algorithm was designed to analyze short audio files with repeating background. To apply it to full songs it required modifications. Liutkus et al. presented Adaptive REPET algorithm for full songs with background changing over time (Liutkus, et al., January 2012).

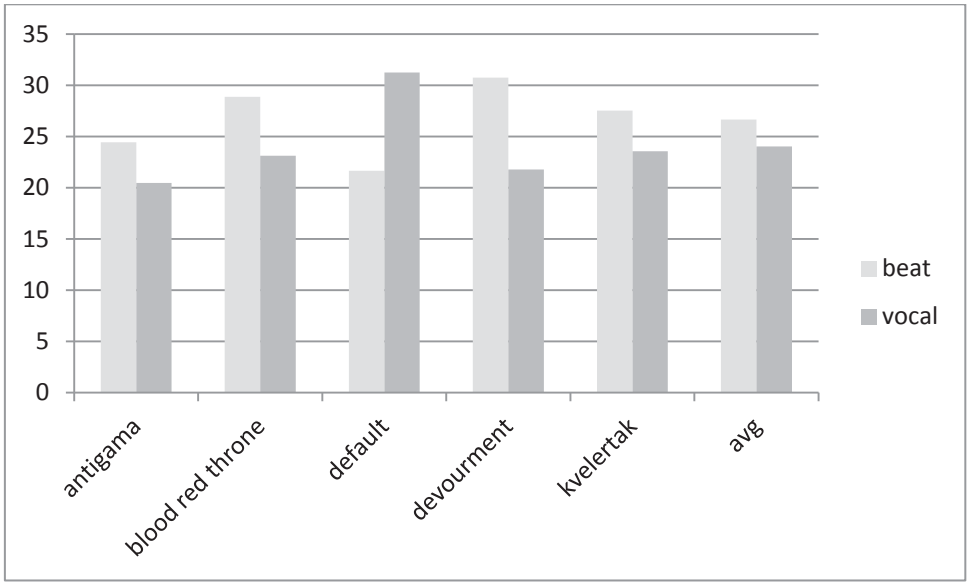
Experiment settings and results

To check how this method may work on rock and metal songs, another experiment was conducted. This time the same songs were used as in original REPET method but in full versions.

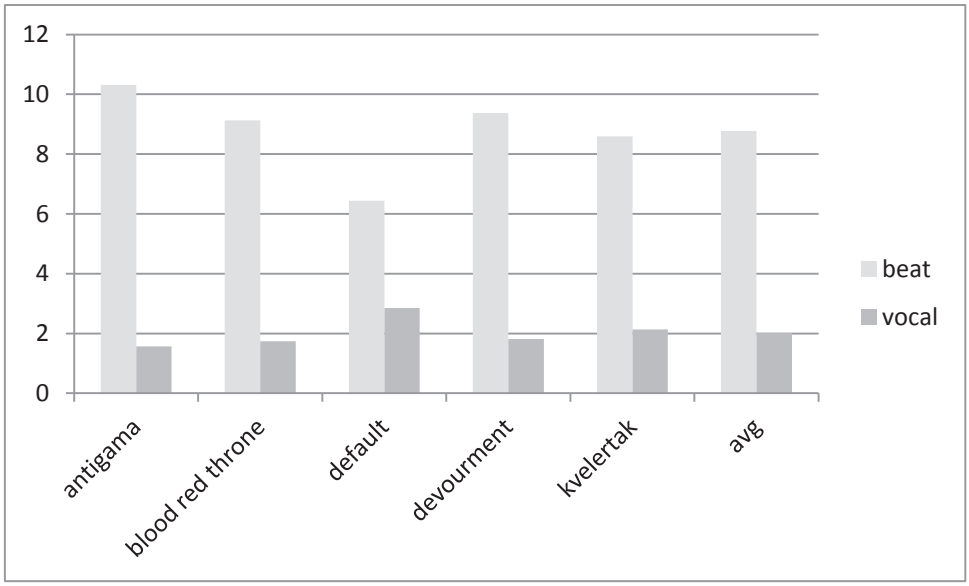
All values were calculated for default Adaptive REPET method settings and then rated using OPS and SDR indicators, same as before. Results are presented in Table 3, Graph 3 and Graph 4.

	Antigama		Blood Red Throne		Default		Devourment		Kvelertak	
	beat	vocal	beat	vocal	beat	vocal	beat	vocal	beat	vocal
OPS	24,4367	20,4604	28,8755	23,1065	21,6597	31,2226	30,7485	21,7903	27,5197	23,5434
SDR	10,3135	1,5707	9,1289	1,7353	6,4339	2,8494	9,376	1,8106	8,589	2,1338

Table 3 OPS and SDR results for Adaptive REPET



Graph 3 OPS results for Adaptive REPET



Graph 4 SDR results for Adaptive REPET

Conclusion

Received results are close to these obtained in original REPET experiment. It must be remembered that original REPET experiment was conducted on parts of songs having non-changing background over time. That means Adaptive REPET method may be successfully used on full songs with background changing over time.

Results vary from each other because of different genre of music in each file. Influence of genre will be examined in the following experiment.

Genre influence - foreground

Experiment settings and results

Second experiment with Adaptive REPET algorithm was set to check how different subgenres of metal and rock music affect results. To do this 3 songs from some subgenres were chosen:

- Black metal
  - Melechesh - Deafeating the Giants
  - Unearthly - Flagellum
  - Vreid - Way Of The Serpent
- Death metal
  - Blood Red Throne - Primitive Killing Machine
  - Broken Hope - The Docking Dead
  - Sphere - Homo Hereticus
- Grindcore
  - Antigama - Crystal Tune
  - Feastem - Dead Eyes
  - Napalm Death - Opposites Repellent
- Rock
  - Clutch - Earth Rocker
  - Queens Of The Stone Age - I Sat By The Ocean
  - The Answer - Somebody Else
- Stoner
  - Black Cowgirl - The Ride
  - Black Space Riders - Louder Than Light Awake
  - Black Tusk - Ender of All
- Thrash metal
  - Angelus Apatrida - Of Men And Tyrants
  - Dublin Death Patrol - Unatural Causes
  - Manic Depression - Impending Collapse

Each song was separated into background and foreground using 40ms windows and default Adaptive REPET settings. Next all tracks were rated using OPS and SDR indicator, same as before and average values for each subgenre were calculated. Results for this experiment are shown in Table 4, Graph 5 and Graph 6.

		Black Metal			AVG
VOCAL	OPS	32,2914	32,6179	37,1047	**34,00467**
	SDR	2,7444	1,7271	1,6326	**2,0347**

		Rock			AVG
VOCAL	OPS	39,1432	45,1447	37,3653	**40,55107**
	SDR	2,1946	3,2377	2,6606	**2,697633**

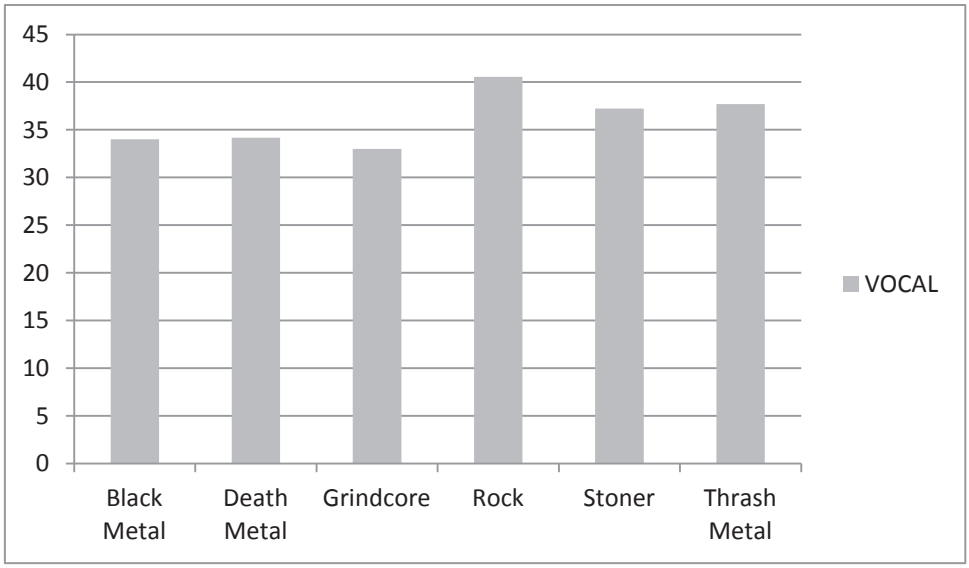
		Death Metal			AVG
VOCAL	OPS	34,5132	35,176	32,8245	**34,17123**
	SDR	2,2135	2,0909	1,5877	**1,964033**

		Stoner			AVG
VOCAL	OPS	42,1388	34,8942	34,6881	**37,24037**
	SDR	2,9473	2,0165	1,4989	**2,154233**

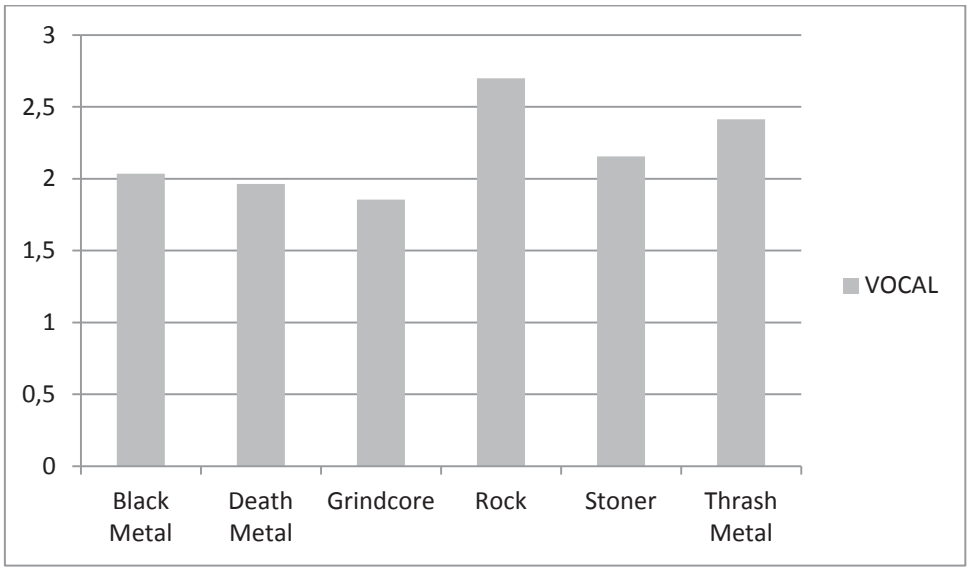
		Grindcore			AVG
VOCAL	OPS	32,2981	33,6103	33,0821	**32,99683**
	SDR	1,4889	1,9337	2,1388	**1,8538**

		Thrash Metal			AVG
VOCAL	OPS	38,4107	39,8734	34,7698	**37,68463**
	SDR	2,073	2,6938	2,4717	**2,412833**

Table 4 Genre influence on foreground separation results



Graph 5 OPS Results



Graph 6 SDR Results

Conclusion

Default window length 40ms in earlier experiments was proven to be very good for separating vocal/foreground from song. Analyzing results presented above conclusion was deduced that Rock is a genre that gives best results. It may appear because of much clearer vocals and less complicated backgrounds comparing to other genres.

Genre influence - background

Experiment settings and results

For last experiment the same database of songs as before was separated using Adaptive REPET method and evaluated using OPS and SDR indicators. This time however window length of 500ms. It was proven before that, this window length gives very good results for background separation. Results for this experiment are presented in Table 5, Graph 7 and Graph 8.

		Black Metal			AVG
BEAT	OPS	60,3096	62,6184	53,5647	**58,8309**
	SDR	7,7999	9,3223	9,0773	**8,733167**

		Rock			AVG
BEAT	OPS	62,2793	50,315	55,3077	**55,96733**
	SDR	8,1622	4,9043	7,2798	**6,7821**

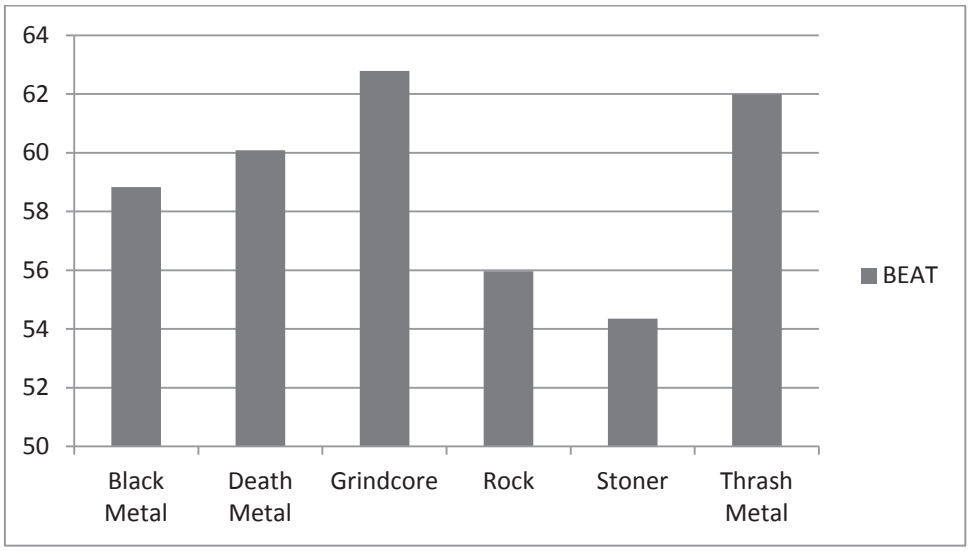
		Death Metal			AVG
BEAT	OPS	60,1852	59,8998	60,1721	**60,0857**
	SDR	8,0308	8,135	9,2584	**8,474733**

		Stoner			AVG
BEAT	OPS	50,031	55,3465	57,6791	**54,3522**
	SDR	5,4804	6,8967	8,9622	**7,1131**

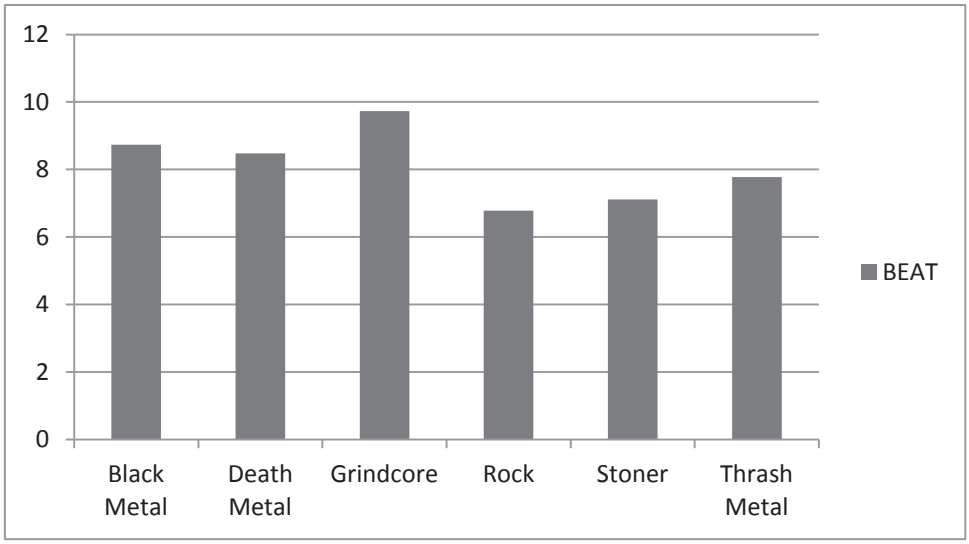
		Grindcore			AVG
BEAT	OPS	59,1219	67,4495	61,7844	**62,78527**
	SDR	10,2274	9,8086	9,1661	**9,734033**

		Thrash Metal			AVG
BEAT	OPS	60,7215	62,1213	63,1129	**61,98523**
	SDR	7,9468	7,6975	7,6838	**7,776033**

Table 5 Genre influence on background separation results



Graph 7 OPS results



Graph 8 SDR results

Conclusion

As proven in first experiment longer window length caused better background separation results. There is bigger variance in results, that means background separation process is more affected by genre of song than foreground separation process. To maximize results every genre would require different settings.

3. AUDIO FINGERPRINTING

Shazam algorithm

Shazam Entertainment, Ltd. presented their own method for audio fingerprinting (Wang, January 2006). It bases on hash tokens calculated for unknown sample and comparing those tokens with previously prepared database.

A. Robust Constellations

First step is to choose feature for recognition. Shazam algorithm was designed for commercial use on mobile phones also. To ensure that signal will survive in presence of noise, distortion and GSM compression spectrogram peaks were chosen. They are robust in the presence of noise and approximate linear superposable. The peak in spectrogram is a candidate if its energy is higher than other points in the region centered on this point. To assure that all chosen points are reasonable uniformly covering spectrogram they are chosen using density criterion. Points with highest amplitude are chosen to ensure they survive all distortions but after that amplitude component is eliminated because its value is no more needed, only coordinates. Obtained scatter plot is called "constellation map" because of its similarity to star constellations.

B. Fast Combinatorial Hashing

The easiest way of comparing constellation map of recorded part to full songs would be sliding over timeline looking for matching points. That solution is simple yet very slow. Shazam presented fast way of indexing constellation maps - fingerprint hashes. First, anchor points are chosen on constellation map. Every anchor point have target zone assigned to itself. Every point in target zone is described using its anchor frequency f_1, its own frequency f_2, time of anchor point t_1 and time difference between points Δt as:

$$Hash{:}time=[f_1{:}f_2{:}\Delta t]{:}t_1$$

The number of hash points is affected by fan-out factor. It specifies number of point assigned to each anchor point in its target zone. Too big fan-out factor makes hashes more space requiring and comparison slower yet more deep.

To create a database for future comparisons above calculations is repeated for each audio file desired to be in database. List of hashes and time offsets is generated for every track.

C. Searching and Scoring

First step in searching for track that match sample sound is repeating hashing process for that sample. Hashes calculated for this sample are compared to those in previously prepared database. When matching value is found, hash, sample hash offset, database track hash offset and track ID are associated. Next thing is calculating difference δt_k between matching sample hash offset t_k and database track hash offset t_k' .

$$\delta t_k = t_k' - t_k$$

Last step is to analyze histogram of obtained differences looking for a peak. Argument of biggest peak will represent offset of recorded part comparing to full song and its value will be number of matching hashes/points.

Overall Experiment settings

All following experiments were performed on the same database containing 3 songs for each chosen subgenre:

- Black metal
  - 1. Melechesh - Deafeating the Giants
  - 2. Unearthly - Flagellum
  - 3. Vreid - Way Of The Serpent
- Death metal
  - 4. Blood Red Throne - Primitive Killing Machine
  - 5. Broken Hope - The Docking Dead
  - 6. Sphere - Homo Hereticus
- Grindcore
  - 7. Antigama - Crystal Tune
  - 8. Feastem - Dead Eyes
  - 9. Napalm Death - Opposites Repellent
- Rock
  - 10. Clutch - Earth Rocker
  - 11. Queens Of The Stone Age - I Sat By The Ocean
  - 12. The Answer - Somebody Else
- Stoner
  - 13. Black Cowgirl - The Ride
  - 14. Black Space Riders - Louder Than Light Awake
  - 15. Black Tusk - Ender of All
- Thrash metal
  - 16. Angelus Apatrida - Of Men And Tyrants
  - 17. Dublin Death Patrol - Unatural Causes
  - 18. Manic Depression - Impending Collapse

Algorithm settings were default if not mentioned otherwise. That means 7 hashes/s when crating database and matching files.

Ideal conditions

Experiment settings and results

First experiment was conducted to check overall efficiency of algorithm when used on modern rock and metal songs. For this experiment songs parts were cut off directly from song files and analyzed by algorithm without using any effects or recording devices.

As the results of this experiment all parts of songs were recognized correctly with high certainly even with short samples and low hash/s ratio.

Good conditions

Experiment settings

For second experiment random 10s samples of all songs in database were recorded when played on speakers connected to computer. All songs were down-mixed to mono for recording time. Mobile phone with Android system was used as recording device. Samples were recorded with mobile phone in front ofone speaker, using call microphone and facing it towards speaker. Sampling rate 44 100 Hz.

Results

Table 6 below presents results of experiment described above. Songs titles and artist were omitted to save space. Instead songs are represented by number corresponding to list above. Columns represent the number of song that was recorded and used for recognition experiment. Rows present number of song automatically recognized from that part. Values represent number of hashes matching recorded part and unmodified song. The bigger the values, the better. Green values mean correct matches, red ones failed.

Part of / Recognized as	song 1	song 2	song 3	song 4	song 5	song 6	song 7	song 8	song 9	song 10	song 11	song 12	song 13	song 14	song 15	song 16	song 17	song 18
song 1	**33**					1	1						1				1	
song 2		**29**									1					1		
song 3	1	1	**35**				1	2	1				1				1	
song 4		1	1	**34**						1	1	1				2		
song 5			1	1	**6**													
song 6	1	1		1		**40**	1		1		1			2				
song 7	1			1		1	**45**		1				1			1		
song 8	1	1	1			1		**25**							1			
song 9									**15**			1			1			
song 10				1		1		1		**29**		1				1	1	
song 11								1	1	1	**31**						1	
song 12								1				**29**						
song 13			1										**55**		1			
song 14							2			1		1	1	**9**				
song 15										1					**18**			
song 16															1	**20**		1
song 17																	**29**	
song 18											2							**20**

Table 6 Good conditions results

Conclusion

All songs were recognized correctly with very high certainty. Average value of 27,89 of matching hashes is a very good result. Other wrongfully matched songs have 1 or 2 matching hashes, that makes results very certainty.

Poor conditions

Previous experiments proved that Shazam algorithm works very good on modern rock and metal songs in ideal and conducive conditions. Following experiments will check how does different poor conditions affect results.

One channel recording

Experiment settings and results

First of experiments conducted in poor conditions was designed the same as the one in good conditions with only one difference. Music played on computer remained in stereo mix and only one speaker was recorded. Table 7 presents results.

Part of / Recognized as	song 1	song 2	song 3	song 4	song 5	song 6	song 7	song 8	song 9	song 10	song 11	song 12	song 13	song 14	song 15	song 16	song 17	song 18
song 1											1							
song 2	2	7						1			1							
song 3			3		1			1					1					
song 4											1							
song 5					14													
song 6						4		1										
song 7					1		5				1		1					
song 8								17										
song 9									4									
song 10								1										
song 11											13		1					
song 12												8						
song 13													17					
song 14					1													
song 15					1										3			
song 16													1			8		
song 17																	3	
song 18																		9

Table 7 One channel recording results

Conclusion

Results of this experiment show that it is very important to record both channels of mix when fingerprinting. Recording only one channel removed so many sounds that it affected comparison result. 78% of recordings were recognized correctly with average number of matching hashes 8,21 . One of samples was recognized as wrong song but with only 2 hashes matching. This could be omitted using *-matchmincount* or *-matchminprop* option with proper value.

Mobile phone on desk

Experiment settings and results

Next experiment was conducted to check if it is possible to analyze songs playing on speakers with mobile phone laying down on desk, next to computer. Mobile phone was recording with 44 100 Hz and songs were played in stereo. Recorded songs could be barely heard because microphone was facing ceiling. Results for this experiment are presented below in Table 8.

Part of / Recognized as	song 1	song 2	song 3	song 4	song 5	song 6	song 7	song 8	song 9	song 10	song 11	song 12	song 13	song 14	song 15	song 16	song 17	song 18
song 1	3																	1
song 2		7														1		
song 3			3				2											
song 4				8														
song 5																1		
song 6						5												
song 7							5											2
song 8								6										
song 9									2									
song 10										8								
song 11					2						4							
song 12												6				1		
song 13													8					1
song 14								2						5				
song 15															5	1		1
song 16																25		
song 17																	4	
song 18			2															12

Table 8 "Mobile phone on desk" results

Conclusion

94% of songs were recognized with average of 7,08 hashes matched per sample and one record was wrongfully assigned to song. This results prove that it is possible to analyze played songs with fairly high probability without requiring special effort from user. It could be used for automatically finding title of song on radio and subscribing it to websites like Last.fm or publishing on Facebook.

4. TEMPO ESTIMATION

Description of algorithm

The algorithm used to estimation of the tempo is using the comb filters. A comb filter is basically a series of impulses that occur periodically(pulse train) (Scheirer, January 1998). As shown on figure 1.

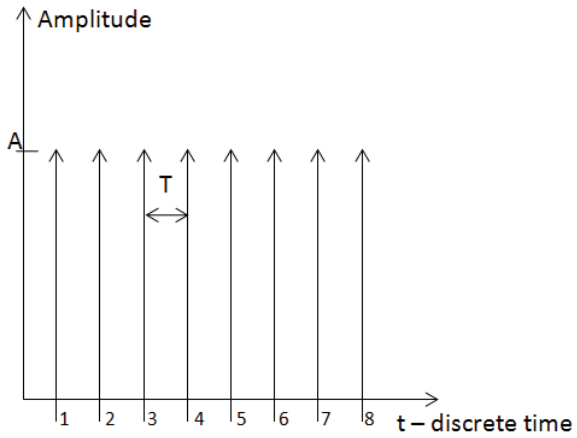


Figure 1 A pulse train, where A is the height of the pulse train; T is the train period

Eric D. Scheirer in "Tempo and beat analysis of acoustic musical signals (1997)" shows that if we assume a comb filter with period *T*, gain *α,* with a pulse train of height *A* and period equal to its delay *k* = *T*. If x_t is the filter input and y_t is the output of the filter, both at discrete time *t*, the equation of the filter is,

$$y_t = \alpha y_{t-T} + (1-\alpha)x_t, \text{and}$$
$$y_0 = (1-\alpha)A$$
$$y_k = \alpha(1-\alpha)A + (1-\alpha)A = (1-\alpha)A(1+\alpha)$$
$$y_{2k} = (1-\alpha)A(\alpha^2 + \alpha + 1)$$
$$\vdots$$
$$y_{nk} = (1-\alpha)A\left(\sum_{i=0}^{n}\alpha^i\right)$$

$$where: y_t - \ filters\ output;\ x_t - \ filters\ input; A - height\ of\ the\ impulse\ train;\ k\ filters - delay;\ \alpha - filters\ gain$$

Equations above shows that:

$$\lim_{n\to\infty} y_{nk} = [(1-\alpha)A]/(1-\alpha) = A$$

If the $T \neq k$, the convergence will go to a smaller value. Also let the λ be the least common multiple (common period) of T and k, using the same logic as above we will get,

$$\lim_{n\to\infty} y_{n\lambda} = \frac{(1-\alpha)A}{1-\alpha^{T/\lambda}}$$

and because $|\alpha|<1$ to get the filter stabile, and $(T/\lambda) \geq 1$,

$$1-\alpha^{T/\lambda} \geq 1-\alpha$$

That means that the filter with delay matching (or evenly dividing) to its period of the filter pulse train, will have a greater output signal then filters with mismatched delay. The frequency domain analysis shows that it is true for any periodic signal. A filter with delay and gain as previous has a magnitude response,

$$\left|H\left(e^{j\omega}\right)\right| = \left|\frac{1-\alpha}{1-\alpha e^{-j\omega T}}\right|$$

The filter has local maxima when $\alpha e^{-j\omega T}$ gets close to 1, in the illustrated filter at the T'th roots of unity, that can be expressed as,

$$e^{-j2\pi n/T}, 0 \leq n < T$$

Due to that ability comb filters can show us the signal with the most matching delay and due to that estimate the signals tempo that will be described in the next section.

For the tempo estimation in music information retrieval we implemented an algorithm in MATLAB based on "Tempo and beat analysis of acoustic musical signals (1997)" by Eric D. Scheirer and "Beat Detection Algorithms (2003)" by Frederic Patin, that use the bank of comb filters approach method. Figure 2 shows an overall view of the tempo analysis procedure.

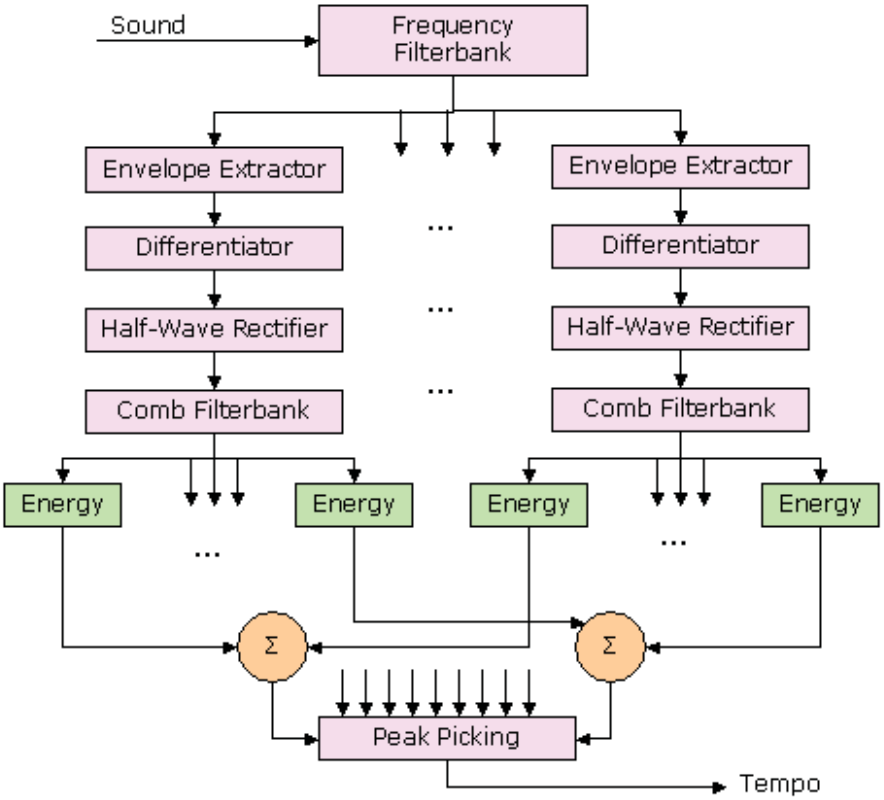


Figure 2 Tempo analysis procedure

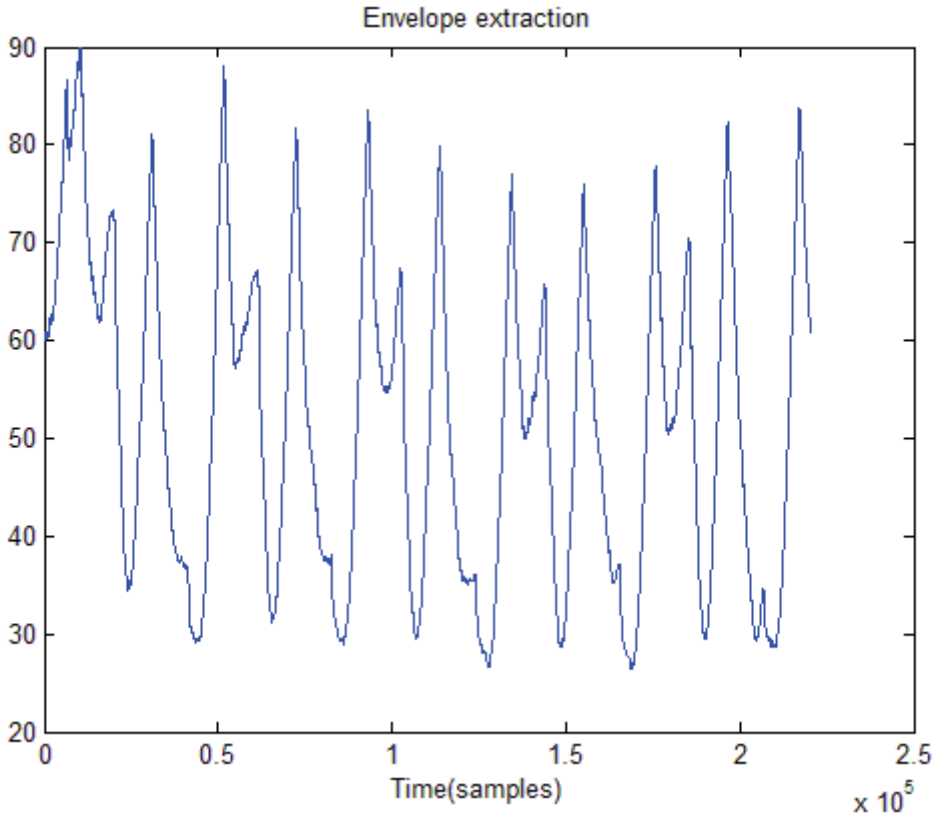
A. Frequency filter bank

At first the input signal is converted to frequency time domain and divided to six frequency bands by the frequency filterbank. The implemented filterbank has six bands with a sharp cutoffs and covers a one-octave range. The filterbank bands are implemented by six filters. For the lowest band is a low-pass filter with cutoff at 200 Hz, four middle bands are archived by band-pass filters with cutoffs at 200 Hz and 400 Hz, 400 Hz and 800 Hz, 800 Hz and 1600 Hz, 1600 Hz and 3200 Hz. The last band is a high-pass filter with 3200 Hz cutoff frequency. Then the signal is converted back to the time domain.

B. Envelope extractor

The next step is the extraction of the envelope from the six band signal. It starts with full-wave rectifying the signal, to decrease high-frequency content and get only the positive part of the envelope. After that signal is convolved with a 200-ms half Hanning window. This kind of a window has a discontinuity at t=0, and smoothly reaches 0 at 200 ms. It has a cutoff at 10 Hz and a low-pass

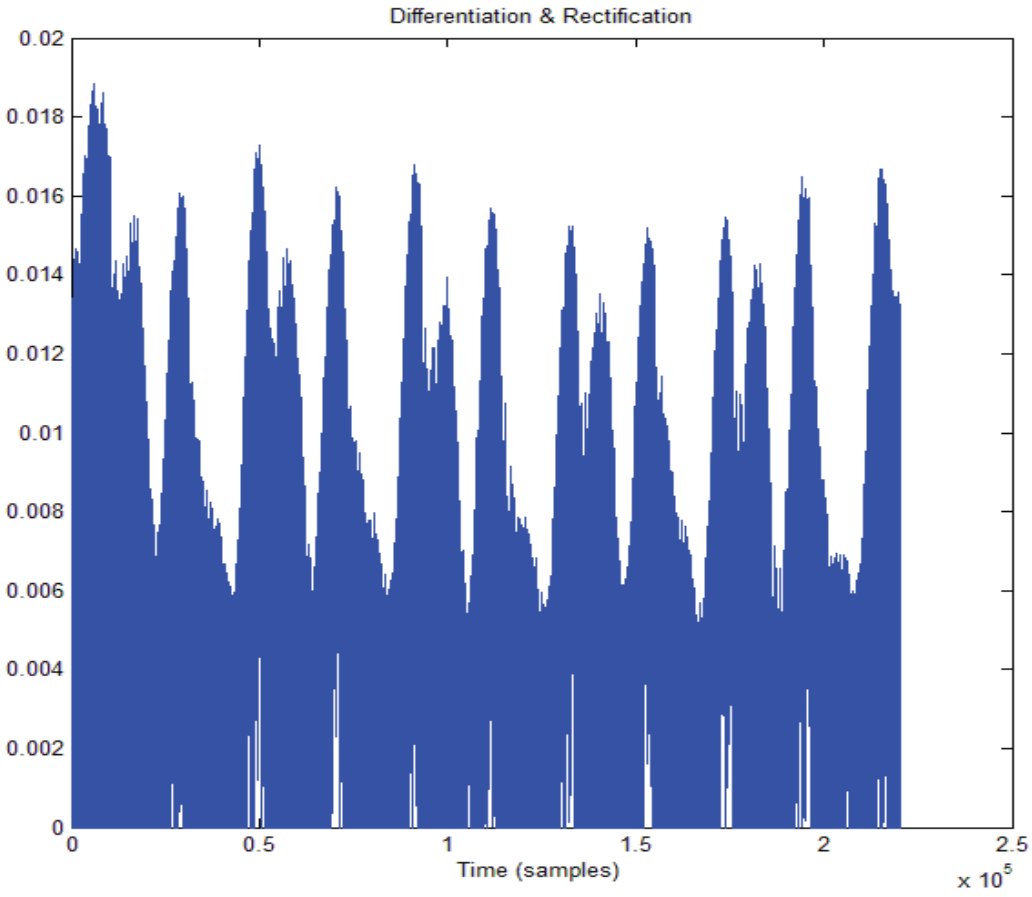
characteristic. Process can be compared to low-pass filtering a signal. The reason to make that with the signal it is because the only information that is really important in tempo estimation is to find sudden changes in the music signal. Witch are more visible after the process. An example of envelope extraction on figure 3 below.



Graph 9 Envelope extraction example

C. Differentiation and half-wave rectification

After the envelope extracting algorithm demands to compute the first order difference function and half-wave rectification. The reason why the signal is differentiated is because the periodic variations of it are more visible in the signals first derivative. Thanks to that we are getting a better view on changes in signal value with are sharp peaks in the envelope. The half wave rectified is made to get only the positive changes in the signal (his increasing energy). We can see an example of the process in next figure 4.



Graph 10 Differentiation and rectification of a signal

D. Comb filtering and estimation of the tempo

When the previous step are complied, the signal is again transformed to frequency domain, and the filterbank of comb filters is used to get the signal energy. As told in previous section the comb filter is a train of impulses that will occur with a known (that we can specify) frequency (tempo), that frequency response to BPM that we want to test (Patin, February 2003) . The frequency is calculated:

$$Comb\ freq(BPM) = \frac{60}{BPM} * fs$$

Where *BPM* is the tested BPM rate and fs is the sampling frequency.

Output gives us the evaluation of the similarity between our song and that impulse train. Due to that we will gain the information what is the BPM rate. What is performed by convolving the banded signal with various number of comb filters (the comb filter bank) and computing the convolution energy that is also the output energy *E*. That energy is computed as follows:

$$E = \sum_{n=0}^{N} [c(n)]^2$$

Where *c* is the convolution of comb filter and the input signal.

The last part is the "Peak Picking". In summed filter bank output there are some peaks where the largest one is the tempo for example 120 bpm, also there will be other smaller peak that will have a harmonic relation such as 3 to 2 of the tempo. The maximum tempo of all of the computed energies is the signal tempo.

Algorithm efficiency

Correctness of the program was tested by computing the tempo of several songs. Also to show how different genders of metal and rock music affect results. To do this 3 songs from some subgenres were chosen:

- Black metal
  - Melechesh - Deafeating the Giants
  - Unearthly - Flagellum
  - Vreid - Way Of The Serpent
- Death metal
  - Blood Red Throne - Primitive Killing Machine
  - Broken Hope - The Docking Dead
  - Sphere - Homo Hereticus
- Grindcore
  - Antigama - Crystal Tune
  - Feastem - Dead Eyes
  - Napalm Death - Opposites Repellent
- Rock
  - Clutch - Earth Rocker
  - Queens Of The Stone Age - I Sat By The Ocean
  - The Answer - Somebody Else
- Thrash metal
  - Angelus Apatrida - Of Men And Tyrants
  - Dublin Death Patrol - Unatural Causes
  - Manic Depression - Impending Collapse

To check if the results are correct tempo was also estimated by MIRtoolbox1.5 (Lartillot, Oliver; Toiviainen, Petri; Eerola,Tuomas, 2008) (Actual tempo in the tables). Tables below show the results of the test (tables 9 to 13):

BLACK METAL			
Artist - Song	Actual Tempo (bmp)	Detected Tempo (bpm)	Error (%)
Melechesh - Deafeting the Giants	170	170	-
Unearthly - Flagellum	115	115	-
Vreid - Way Of The Serpent	169	168	0,59
		Average Error	0,20

Table 9 Tempo test for Black metal.

DEATH METAL			
Artist - Song	Actual Tempo (bmp)	Detected Tempo (bpm)	Error (%)
Blood Red Throne - Primitive Killing Machine	135	136	0,74
Broken Hope - The Docking Dead	136	133	2,21
Sphere - Homo Hereticus	160	160	-
		Average Error	0,98

Table 10 Tempo test for Death metal.

GRINDCORE			
Artist - Song	Actual Tempo (bmp)	Detected Tempo (bpm)	Error (%)
Antigama - Crystal Tune	121	115	4,96
Feastem - Dead Eyes	155	151	2,58
Napal Death - Opposites Repellent	124	120	3,23
		Average Error	3,59

Table 11 Tempo test for grindcore.

THRASH METAL			
Artist - Song	Actual Tempo (bmp)	Detected Tempo (bpm)	Error (%)
Angelus Apatrida - Clockwork	120	80	33,33
Dublin Death Patrol - Unatural Causes	147	143	2,72
Manic Depression - Impending Collapse	109	108	0,92
		Average Error	12,32

Table 12 Tempo test for Thrash metal.

ROCK			
Artist - Song	Actual Tempo (bmp)	Detected Tempo (bpm)	Error
Clutch - Earth Rocker	160	162	1,25
QOTSA - I Sat By The Ocean	116	116	-
The Answer - Somebody Else	134	135	0,75
		Average Error	0,67

Table 13 Tempo test for rock.

Conclusion

The tempo detection works very well witch error almost never being higher that 3 %, in fact most of the time the average error is below 1%.

Program work specially good with gander that have strong or characteristic drum section, those are black metal, death metal and rock. Specially in black metal where there is almost no error.

The error increases in genders with are characterized by highly complex compositions such as grindcore, we also can see one high error (33%) in thrash metal song, but beside of that, performance in this gender is also high.

5. CHORD RECOGNITION

Pitch Class Profiling

Pitch class profile(PCP) is a vector that shows how the frequency content of a chord sound corresponds to the 12 notes (semi-tones) of a standard chromatic scale (Osmalskyj, J.; J.-J., Embrechts; Van Droogenbroeck, M.; Pierard, S., May 2012). In other words the PCP shows the distribution of energy of the chord. Tones are defined in logarithmic scale and the energy is expressed in natural units. Pitch class profile was first presented by Takuya Fujishima (Stanford University) in "Real-time Chord Recognition of music sound".

Here are the steps to obtain the 12-bin PCP. First the input sound is transformed to a discrete time Fourier transform (DFT) spectrum *X(N)*. After that each of the frequency bins of the spectrum are mapped. Mapping could be compared to dividing the signals spectrum to regions. For example the frequencies like 430 Hz or 433 Hz are mapped to semi-tone A at 440 Hz. The M(k) function is the mapping of the frequency bins to PCP.

$$M(k) = \left| 12 \times log_2 \left(\frac{k}{N} \times \frac{f_s}{f_{ref}} \right) \right| mod\ 12$$

Each of the bins (k) are mapped to closest semi-tone (note) .Where *N* is the number of DFT bins, k is a bin in the DFT where 0≤ k ≤ N-1, the f_s is the sampling rate and the f_{ref} is the reference frequency that corresponds to *PCP(0).*

Because in most cases the DFT bins do not match the notes frequencies that is why ti is necessary to calculate the "distance" form the semi-tone to DFT frequency. The distance value varies from 0 to 1 where 0 is the exact same value of DFT and semi-tone frequency and 1 is then the DFT is exactly between two notes. Is calculated as follows

$$distance = 2 \times abs \left(\frac{12 \times log\left(\frac{[Hz_{bin}]}{440}\right)}{\log(2)} - \frac{\left(12 \times log\frac{[Hz_{note}]}{440}\right)}{\log(2)} \right)$$

$$where: Hz_{bin} - freguency\ of\ DFT\ bin;\ Hz_{note} - freguency\ of\ closest\ note;$$

To finally compute the PCP the summation of the PCP elements need to be weighted using the distance to closest semi-tone. There is a number of functions that are used to weight the PCP. The weighting distance functions are: uniform, discrete, linear, anti-quadratic, exponential, and qaussian. The uniform one causes that the distance do not affect the result. The most used method is the discrete one due to that one only the frequencies that are inside the narrower region are considered. In other ones, the weight of the distance will decrease as it will go farther from the semi-tone frequency (Cabral, Giordano; Briton, Jean Pierre; Pachet, Francois, 2005).

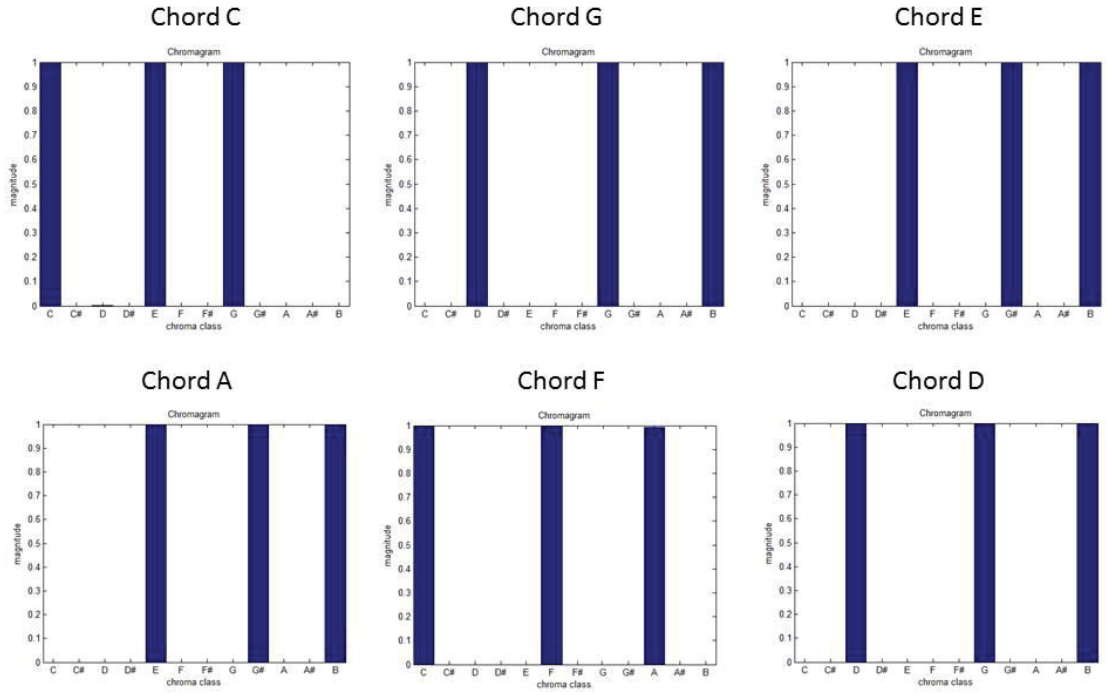
$$PCP_{[p]} = \frac{\sum_{k:M(k)=p}|X(k)|^2 \times f(distance)}{\sum_{k:M(k)=p} f(distance)}$$
$$where: f(distance) \; - \; the\ weighting\ function$$

The PCP is computed but it needs to be normalized. It is necessary to normalize the values because the chord can be played louder or softer than others. Normalization is obtain as

$$normalized\_PCP(p) = \frac{PCP(p)}{\sum_{j=0}^{11} PCP(j)}$$

Where p is the index of the normalized PCP bin $0 \leq p \leq 11$.

The ideal PCP representation of 6 chords are shown in figure 5.



Graph 11 The ideal representation of 6 chords

Neural Network

Neural network in an computational model inspired by central nervous system (the brain). That is able to machine learning and pattern recognition. It is a system of interconnected “neurons” that compute the output by feeding the inputs through the network (Opaliński, December 2012).

The network neuron is a mathematical function conceived as a crude model, or abstraction of biological neurons. Such a neuron receives one or more inputs then sums them to produce an output . In most of the cases the inputs are weighed, and the sum is passed through a non-linear function known as the activation function. For example let the neuron has *m*+1 inputs with signals x_0 to x_m and weights w_o to w_m . Usually the input $x_0$is assigned to the value 1, that makes it a bias input with $w_{k0} = b_k$. There is only *m* actual inputs to the neuron. The output of *k*th neuron is

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj}x_j\right)$$

where φ is the activation function. Figure 6 shows the neuron of a neural network.

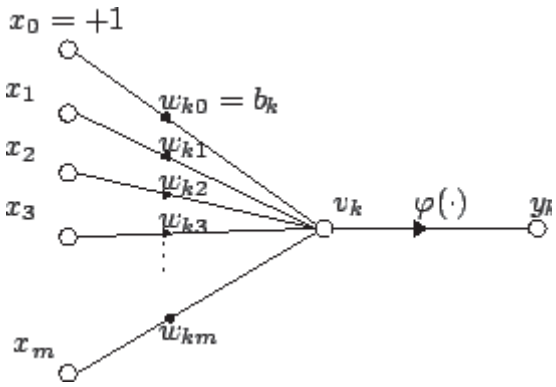


Figure 3 Neuron of a neural network

The output of the neuron y_k, becomse the input for another neuron in the network. It can also becomes the part of the output vector.

A good example of using a neural network is recognition of handwritten letters. In such network the input neurons are activated by the pixels of the input image of the letters. Then during the training process the activation of each neuron is weighted and transformed, by functions set by the network designer, until the output neuron is activated by the correct letter.

There is a various number of tasks that are hard to solve using ordinary rule-based programming there neural network have great performance, such as chord recognition.

Experimental settings and results

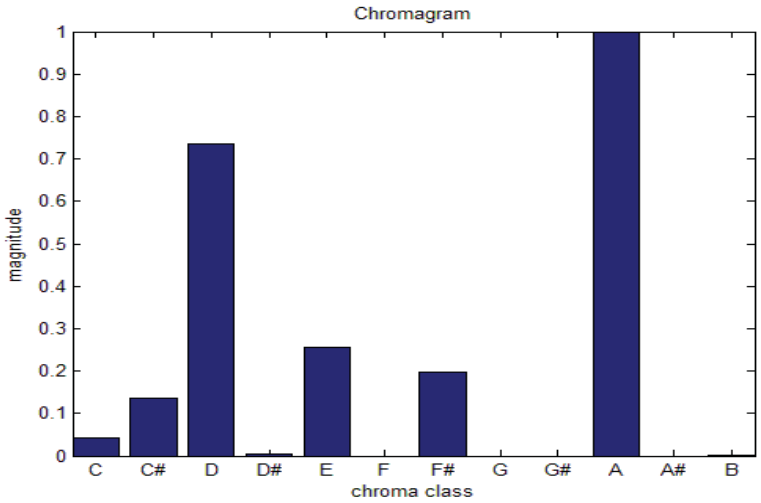
There is a great number of chord because of various number of music types in the world. Because of that the recognition was limited to 6 most common chord in western music. Those are:

C, G, E, A, F, D

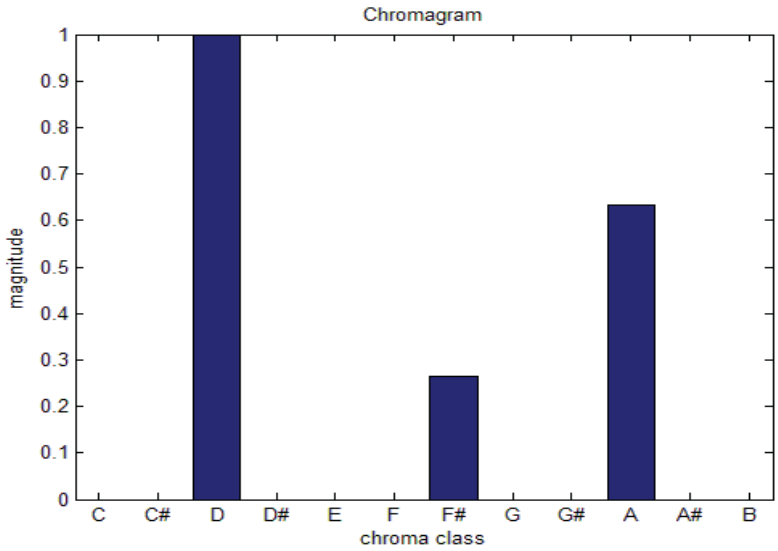
All of the PCP vectors are stored in a unique (excel) file, which is organized as follows, every line consist a 12-element that correspond to the semi-tones. These were computed using MIRtoolbox 1.5 for Matlab (Lartillot, Oliver; Toiviainen, Petri; Eerola,Tuomas, 2008).

The chord dataset used in chord recognition is the dataset created by J. Osmalsky, J-J. Embrechts, S. Pierard, M. Van Drooogenbroeck in "Neural network for music chords recognition". Chords are played by an acoustic guitar, because it is probably the most common instrument to play chord in western music. Half of them are played and recorded in an anechoic chamber using a wideband microphone (01 dB MCE320). The second half of the chord was recorded in a noisy environment using a live microphone (Shure SM58). All of the chord are recorded in the WAV format, sampled at f_s = 44100 Hz, and quantized with 16 bits.

The difference between the energy distribution in noisy and noise-free chord is show in the graphs 9 and 10.



Graph 12 A chromagram of noisy D chord



Graph 13 A chromagram of a noise-free D chord

Chord recognition was realized by using Matlabs neural network toolbox and creating a pattern recognition neural network. The network architecture (shown on figure 7) is as follows, there are 12 corresponding to the semi-tones input attributes. Network has two layers, hidden layers and output layer. The hidden layer has 35 neurons, the output has 6 neurons. The output of the neural network in a 6 element vector which correspond to output neuron witch correspond to detected chord.

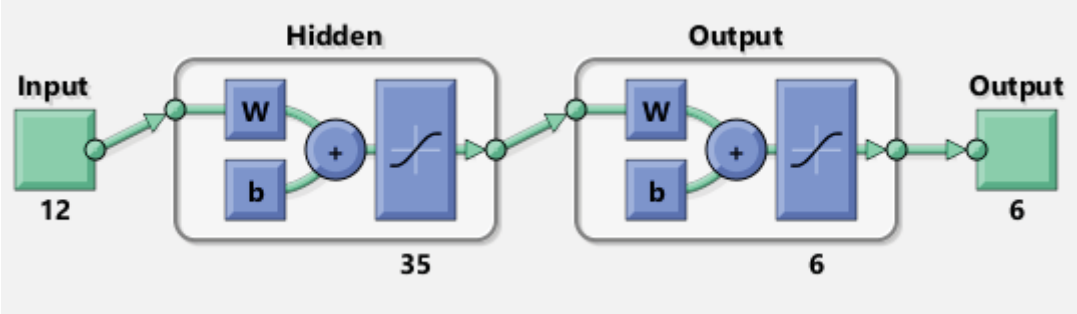
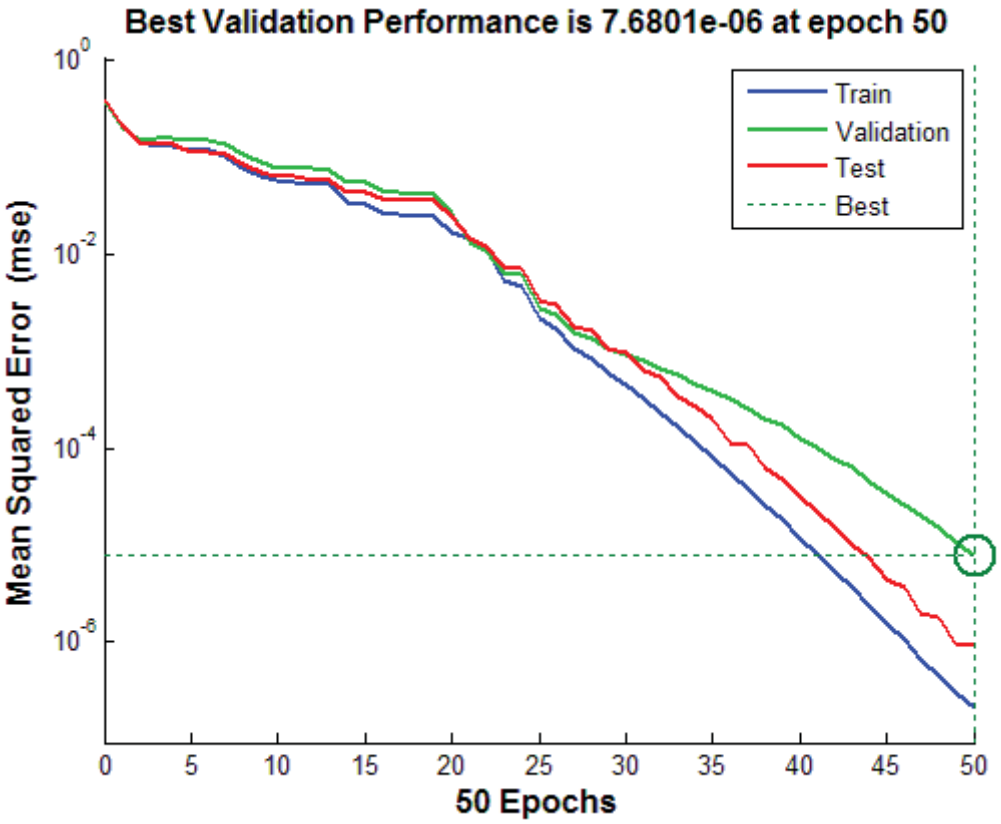


Figure4 Neural network architecture

The dataset that is used to train and test the performers of the network is described above. The network was trained with both noisy and noise-free chords. The number of chords is 60 for each class that is 360 chords, network was trained using 252 chords both noisy and noise-free ones. Figures 8 and 9 show the network performance.



Graph 14 Performance of the neural network

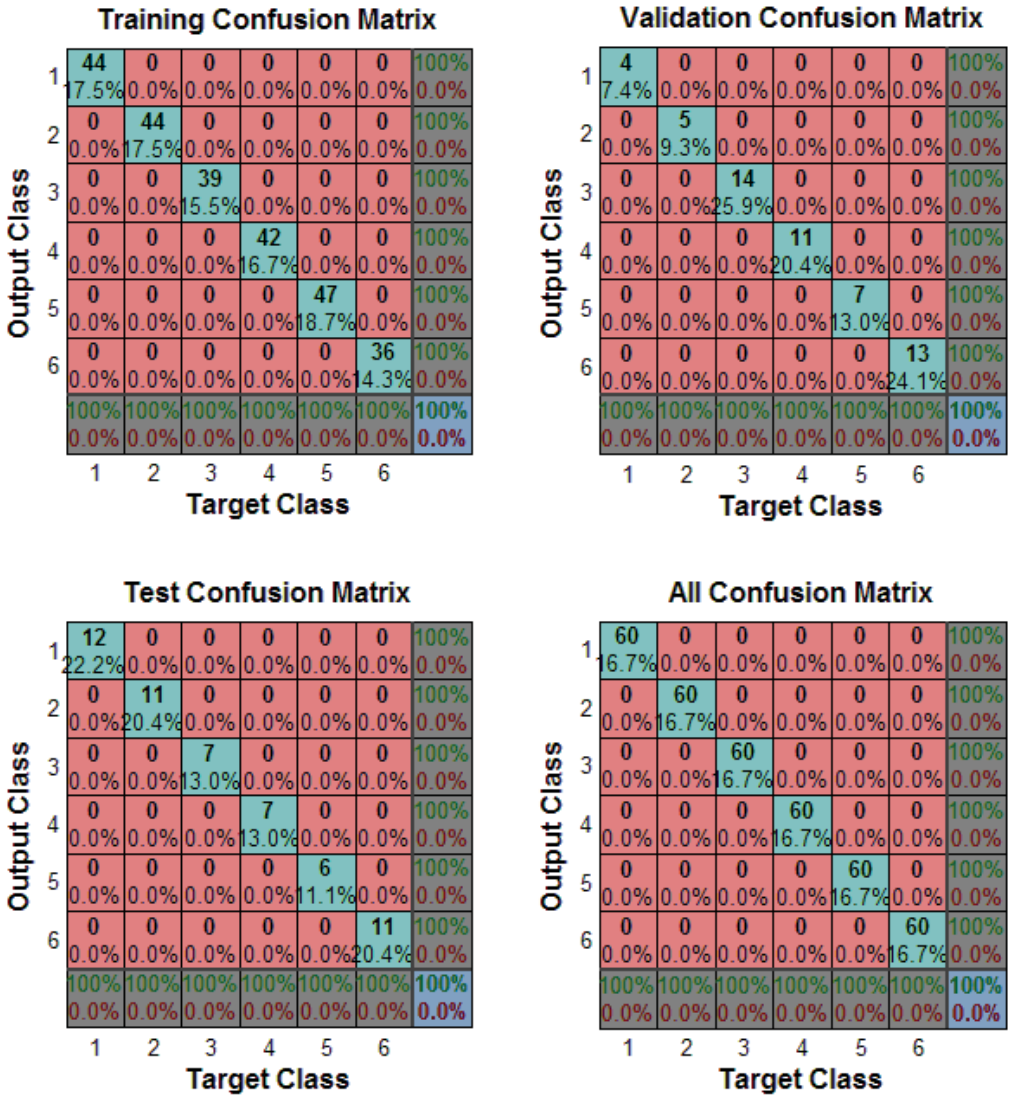


Figure 5 Matrices of networks confusion

Reading the confusion matrixes. For example in test confusion matrix and the first column assigned to target class 1. There are 12 class 1 targets assigned correctly (GREEN) to output class 1, none was assigned incorrectly to other classes (RED). The grey column and row shows the percentage of correct and in correct assignment. The percentage in the green cells shows the size of the targets overall . For example the second cell in the second column is 20,4% because the number of all targets for all classes is 54 and the number of targets in cell is 11 so $100\% * 11 \div 54 = 20{,}4\%$.

Conclusion

As the graph 11 and figure 8 shows the neural network managed to recognize the chord that it was tested whit. After 50 epochs it was able to recognize chord with no mistake that shows that, is a great tool to perform this task. Also that neural network is a great way to replace more complex and have bigger memory usage.

6. CONCLUSION

During thesis work some of popular and overall effective methods for audio separation, audio fingerprinting, tempo estimation and chords recognition were evaluated on examples of modern rock and metal music. It turned out that default values on most of above methods are ineffective when working on this kind of music because of its accumulation and complexity.

In background/foreground separation part it was proven that windows length has influence on results. It should be chosen accordingly to objective. Longer window lengths give better background but weak foreground separation and contrariwise. Genre influence experiments showed that type of processed music has also impact on background separation results, foreground separation results were less affected. Summarizing, it is possible to separate modern rock and metal songs into background and foreground with quite good efficiency using correct settings.

Audio fingerprinting experiments indicated that it is very important to record for analysis in best possible conditions. Poorly recorded parts could not be recognized correctly even with very high hashes/sec ratio. Last experiment in audio fingerprinting chapter proved that modern mobile phones are sufficient to record and analyze songs with high accuracy without users effort.

Tempo estimation shown that this task metal music can be difficult because of complexity of this music. In the results the error very visible in genders with rapidly changing tempo such as thrash metal or grindcore that had nearly five times bigger average error then much simpler gender like rock or black metal.

In chord recognizing the most important is a good quality dataset. Of course it has possible to recognized the chord recorded in noisy room, but still it was done using a good quality microphone. Also good performance was possible mostly thanks to noise-free recordings used to train the network.

7. FUTURE WORK

Conducted experiments and obtained results show that Music Information Retrieval field accomplished a lot though last years. Examined algorithms perform well even on such complicated genres of music as modern rock and metal when using correct settings.

Future work might consist similar experiments conducted on big databases with this kind of music. Next step should be comparison of evaluated algorithms with new solutions presented after this thesis to find most effective ways of analysis for modern rock and metal music.

8. REFERENCES

Aucouturier, Jean Julien and Bigand, Emmanuel. 2012. *Mel Cepstrum & Ann Ova: The Difficult Dialog Between MIR and Music Cognition.* 2012.

Cabral, Giordano; Briton, Jean Pierre; Pachet, Francois. 2005. *Impact of Distance in Pitch Class Profile Computation.* Paris : Sony Computer Science Lab, 2005.

Downie, J. Stephen. summer 2004. The Scientific Evaluation of Music Information Retrieval Systems: Foundation and Future. *Computer Music Journal 28/2.* s.l. : MIT Press, summer 2004, pp. 12-23.

Emiya, Valentin, et al. February 2011. Subjective and objective quality assessment of audio source separation. *IEEE Transactions on Audio, Speech and Language Processing.* February 2011.

Fujishima, Takuya. 1999. *Real-time Chord Recognition of music sound.* Stanford, USA : Stanford University, 1999.

Goto, Masataka and Goto, Takayuki. September 2005. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. *Proceedings of the ISMIR 2005.* September 2005.

Haitsma, Jaap and Kalker, Ton. August 2002. *A Highly Robust Audio Fingerprinting System.* August 2002.

Jang, Dalwon, et al. June 2006. *Automatic Commercial Monitoring for TV.* June 2006.

Kassler, Michael. spring-summer 1966. Toward Music Information Retrieval. *Perspectives of New Music.* spring-summer 1966, pp. 59-67.

Kozek, Werner and Pfander, Götz. July 2007. *Time-Frequency Analysis: Tutorial.* July 2007.

Lartillot, Oliver, Toiviainen, Petri and Eerola, Tuomas. 2008. A Matlab Toolbox for Music Information Retrieva, *Data Analysis Machine Learning and Applications,Studies in Classification, Data Analysis. and Knowledge Organization, Springer-Verlag, 2008*.

Lamere, Paul. November 2008. Social Tagging and Music Information Retrieval. *Journal of New Music Research 37 (2).* November 2008, pp. 101-114.

Liutkus, Antoine, et al. January 2012. *Adaptive Filtering for Music/Voice Separation Exploiting the Repeating Musical Structure.* January 2012.

Mulder, D.G.J. July 2014. *Automatic Classification of Heavy Metal Music.* July 2014.

Neves, Cláudio, et al. March 2009. *Audio Fingerprinting System for Broadcast Streams.* March 2009.

Opaliński, Artur. December 2012. *Sieci neuronowe.* Gdansk : Lecture on Neural Networks Gdansk University of Technology, December 2012.

Osmalskyj, J.; J.-J., Embrechts; Van Droogenbroeck, M.; Pierard, S. May 2012. Neural networks for Musician Chords Recognition. *Journees d'informatique misicale.* May 2012.

Pampalk, Elias, Dixon, Simon and Widmer, Gerhard. August 2003. *Exploring Music Collections by Browsing Different Views.* August 2003.

Patin, Frédéric. February 2003. *Beat Detection Algorithms.* February 2003.

Rafii, Zafar and Pardo, Bryan. January 2013. REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation. *IEEE Transactions on Audio, Speech and Language Processing vol. 21.* January 2013, pp. 73-84.

Scheirer, Eric D. January 1998. Tempo and Beat Analysis of Acoustic Musical Signals. *J. Acoust. Soc. Am. 103 (1).* January 1998, pp. 588-601.

Shrestha, Prarthana and Kalker, Ton. August 2004. *Audio Fingerprinting in Peer-to-Peer Networks.* August 2004.

Tsatsishvili, Valeri. November 2011. *Automatic Subgenre Classification of Heavy Metal Music.* November 2011.

Vignoli, Fabio, Gulik, R.V. and Wetering, H.V.D. August 2004. Mapping Music In The Palm Of Your Hand, Explore And Discover Your Collection. *Proceedings of the ISMIR 2004.* August 2004.

Vincent, Emmanuel. March 2012. Improved perceptual metrics for the evaluation of audio source separation. *10th Int. Conf. on Latent Variable Analysis and Signal Separation (LVA/ICA).* March 2012.

Wang, Avery Li-Chun. January 2006. *An Industrial-Strength Audio Search Algorithm.* s.l. : Shazam Entertainment, Ltd., January 2006.

Wiering, Frans. December 2006. *Can Humans Benefit from Music Information Retrieval?* December 2006.