Automatic Extraction of Drum Sounds from Music Recordings

Toningenieur-Projekt

Gabriel Hülser

Supervisor: Dr. Alois Sontacchi

Graz, June 8, 2017

kunst uni graz

institut für elektronische musik und akustik

Abstract

The aim of this work is to implement an algorithm that automatically locates, extracts and classifies isolated drum sounds from music recordings. The use of drum sounds taken from external music recordings is a wide spread technique in modern popular music, also called sampling. For musicians locating and slicing these relatively short sound snippets can be very time consuming, especially when working with big music collections. Therefore, such a system could help musicians by making it easier to create drum libraries from their personal music collections.

The procedure to establish the considered algorithm can roughly be divided into four stages. In the first stage potential sections in the waveform are selected using crest factor and root mean square and in the second stage beat tracking is used to segment the sections into individual sounds. Subsequently the sounds are classified into the categories non percussive/harmonic, kick drum, snare and hi-hat. For the classification step a set of temporal and spectral features are calculated and reduced in dimensionality by using linear disciminant analysis. For the actual classification maximum likelihood method is used. Labeled training data was gathered from several drum libraries and breakbeat compilations. An overall classification rate of 89% was achieved, which may be further improved by using more training data.

Contents

1 Introduction: Sampling in Music

In the context of music production the term sampling refers to the act of taking a piece from a music recording to create instruments or sound effects for the purpose of reusing it in a different piece of music. Although this technique was already used by experimental musicians in the 1940's it became very popular among hip hop artists in the late 1970's and 1980's [Ess]. At the early stages of hip hop it was common to use loops from funk, soul or disco music spanning one or multiple bars to give rappers or b-boys a backing track for their performance. After a while the musicians started to record single drum sounds from popular break beats into their samplers to create their own drum kits and rhythm tracks. To this day this practise is very common among artists in popular and especially electronic music.
While there exist many sample libraries compiling drum sounds from the most popular break beats it is still part of many musician's work to search for recordings containing isolated drum sounds and to create their personal sound collections by cutting them out and combining them to new sounds. In the early stages of sampling, meaning the 1990's and early 2000's, the samples were cut almost exclusively from vinyl records since they were more affordable than CD's and a lot of the source material was not very popular and therefore simply not available on digital media. Although many musicians nowadays claim to still use vinyl records, may it be due to tradition or aesthetical reasons, It is obvious that digital recordings are widely used for sampling, simply because of their availabilty from the world wide web.
While the use of physical recordings forces the musicians to listen through the whole piece to find the sounds they are looking for, the advantage of digital music is that music information retrieval algorithms can be used to speed up the process of finding samples or even to completely undertake it.
The goal of this work is to take a first step into this direction and to implement an algorithm that automatically locates, extracts and classifies isolated drum sounds from music recordings.
Perfecto Herrera et. al examined different techniques for the classification of drum sounds achieving high hit-rates up to 97% [HYYG02]. While there are multiple works on the extraction of drum sounds from polyphonic music recordings, which could be considered a source seperation task [ZPDG02], the author is not aware of any work addressing the problem of locating isolated drum sounds in music recordings.

Structure of this work Chapter 2 discusses the established method including a descriptions of the Pre-Selection 2.2 and Segmentation step 2.3 as well as an overview of the used features 2.4, the dataset 2.5 and finally the classification method 2.6.
In section 3 the results are presented and discussed and section 4 presents the conclusion of this work.

2 Method

2.1 Overview

Figure 1 shows an overview of the algorithm, which can be roughly divided into four stages. The aim of the first stage is to select parts from the input signal that have a high potential to be percussive sounds. This is done by using low-level-features and thresholding. In the second step, these parts are segmented into single hits using an onset detection algorithm. Subsequently a feature vector is calculated for each of the segments and a classifier is used to identify drum sounds and divide them into the categories kick, snare and hi-hat.

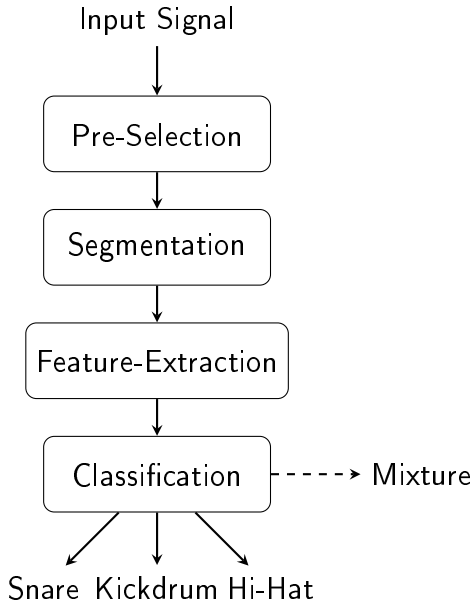

Figure 1 – Overview of the system

2.2 Pre-Selection

The principal goal of this step is to find sections within the waveform, that have a high potential to exhibit isolated drums to reduce computation time. Since feature extraction for a typical input signal of about 3 minutes length would take a high computational effort, relevant segments are segregated from the input signal beforehand by using a binary mask. Figure 2 roughly describes the process of calculating the binary mask.

The pre-selection is based on root mean square and crest factor of the signal. Assuming

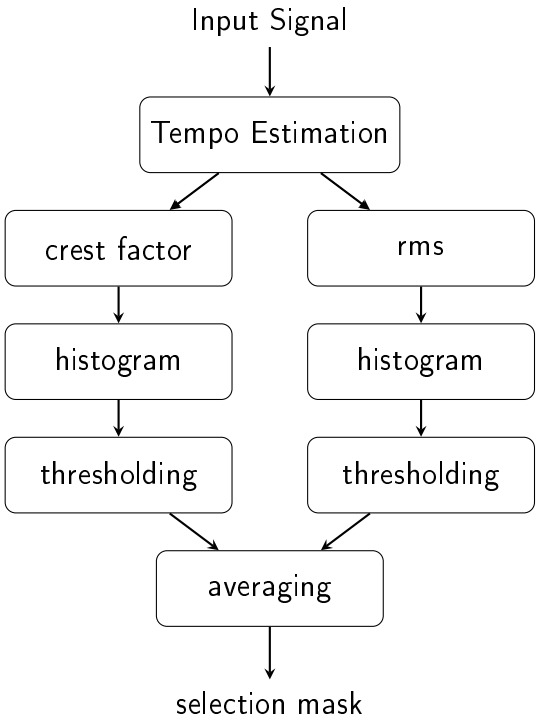

Figure 2 – Outline of the pre-selection step

four beats per bar, the length of the analysis windows are chosen so that they span one bar for the crest factor and half a bar for the root mean square while the hop-size is fixed at a length of 100 ms. For tempo estimation a beat tracking algorithm by Dan Ellis is used [Ell07]. Crest factor and rms are then binarized by separate threshold values. To determine the threshold values the histograms of crest factor and rms are calculated. For the crest factor the threshold is defined by steepest descent in the upper half of the histogram. Similarly the threshold for the rms is derived from the slope in the lower third of the histogram (figure 3).

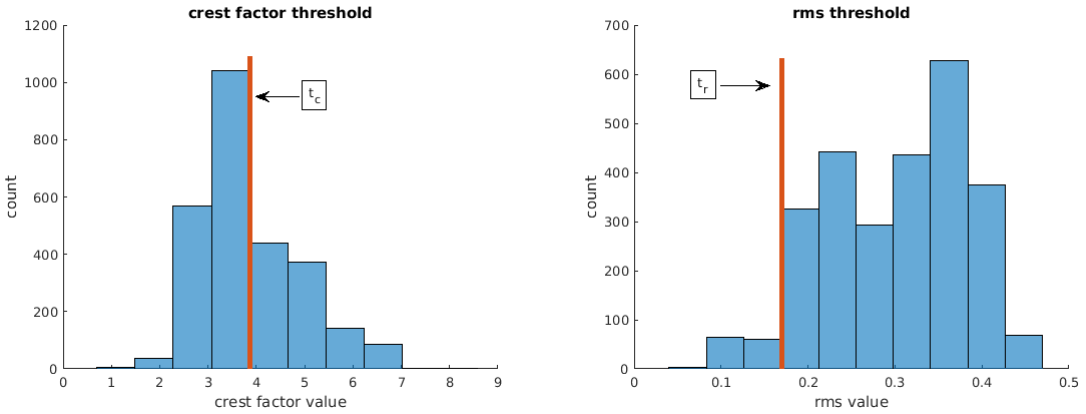

Figure 3 – Thresholds for crest factor and rms

The crest factor is binarized by setting every value smaller than the threshold t_c to zero.

$$B_c = \begin{cases} 0, & \text{if } x_c < t_c \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

Analogously every value of the rms higher than the threshhold t_r is set to zero.

$$B_r = \begin{cases} 1, & \text{if } x_r < t_r \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Finally the binarized functions of rms and crest factor are added together and averaged by a time window of 5 frames.

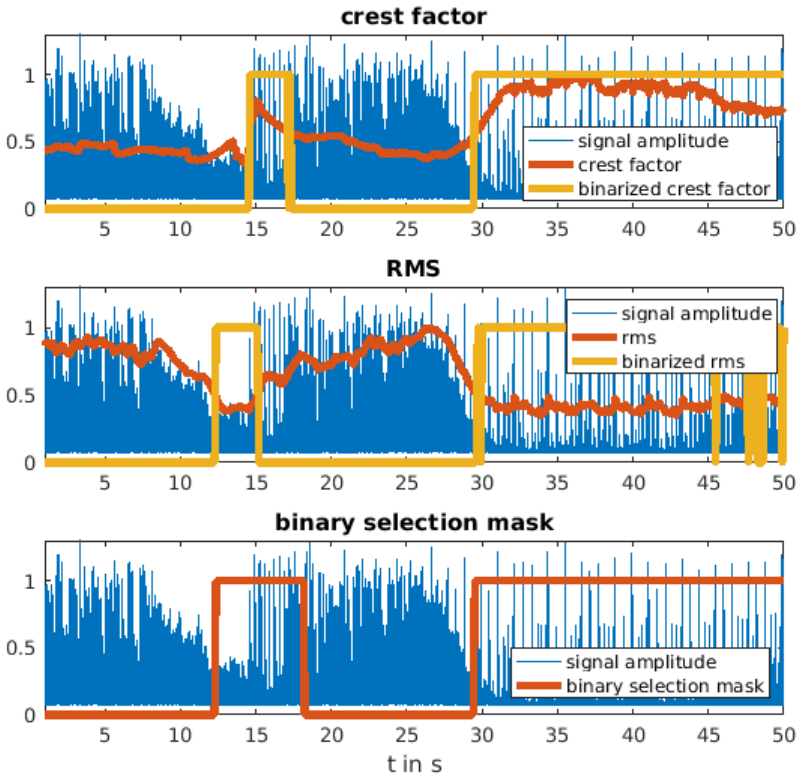

Figure 4 – Calculation of the selection mask. Crest factor before and after thresholding (top), RMS before and after thresholding (middle) and combined selection mask (bottom).

2.3 Segmentation

The previously determined parts are now cut into single segments, ideally consisting of single sounds. Therefore an onset detection is performed and the signal is cut at the found onsets. To ensure a clean cut and to avoid any clicks at the beginning the start times are moved to their nearest zero crossings. For the implementation of the onset detection an extended version of the spectral flux log filtered by Böck et. al is used here [BKS]. On this detection function simple peak picking with a minimum peak distance of a demisemiquaver is applied to estimate the relevant onsets.

2.4 Features

The selection of the features is partly based on the work of Perfecto Herrera et al. [HYYG02]. In general the features can be divided into two groups namely temporal and spectral features.

2.4.1 Pre-processing

Signal Envelope The signal envelope is obtained by using a N-tap Hilbert filter, where N is chosen to represent a time window of 100 ms (MATLAB: `envelope(x, N)`. Subsequently the result is lowpass filtered by a 6-th order butterworth filter with a cut-off frequency at 100 Hz.

Attack Time Estimation For some features it is distinguished between the attack and the decay phase of the signal and therefore the attack time is estimated as a first step. For the actual implementation, the weakest effort method, as proposed by Peeters [Pee04], is used.
Then as depicted in figure 6, ten equally spaced threshold values Θ_i are defined from the global maximum of the envelope. The time t_i where the envelope first reaches the corresponding threshold Θ_i is then used to define the "effort" as the interval between successive time instances $\omega_i = t_{i+1} - t_i$. Then the average value $\bar{\omega}$ over the efforts is calculated and the threshold th_{st} of the starting time of the attack phase is defined as the first threshold where the effort goes below $K * \bar{\omega}$. Similarly the last threshold for which the effort goes below $K * \bar{\omega}$ is taken as the end of the attack phase th_{end}. For the constant K a value of 3 is used for the implementation. The start time t_{st} and end time t_{end} are then set to the local maxima around the corresponding thresholds th_{st} and th_{end}.

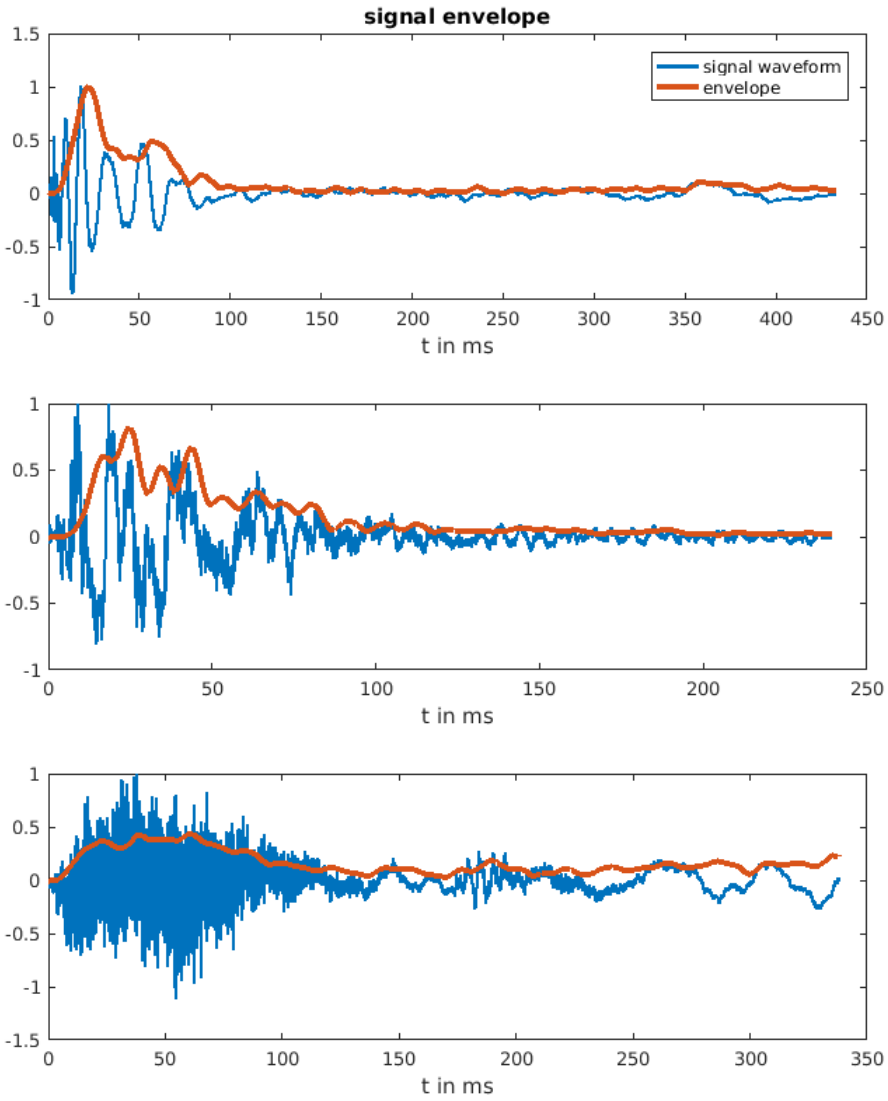

Figure 5 – Signal $x(t)$ (blue) and its envelope $e_x(t)$ (red) for three different exemplary sounds. Kick drum (top), snare (middle) and hi-hat (bottom).

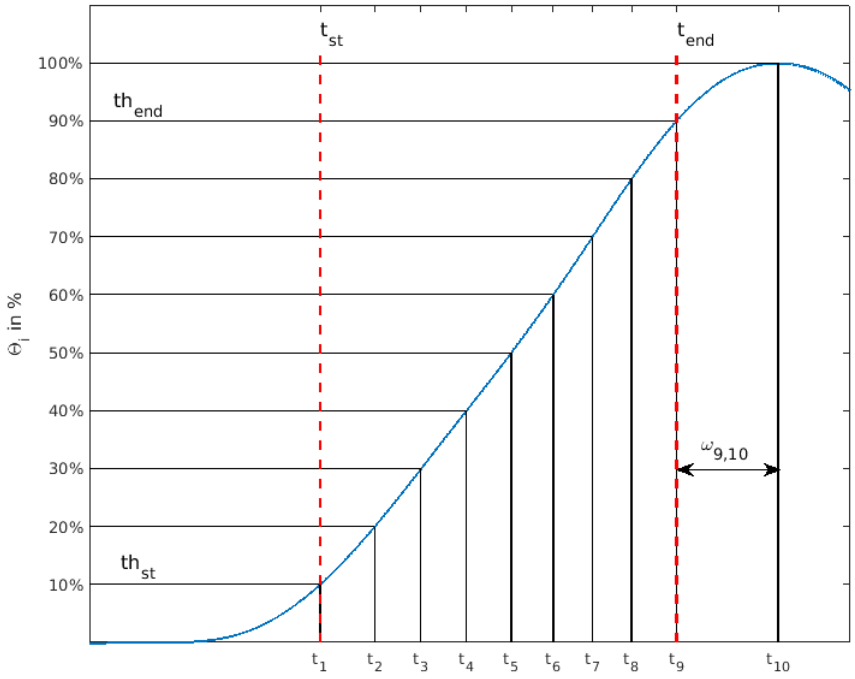

Figure 6 – Weakest effort method for attack time estimation according to Peeters [Pee04]. Start time (t_{st}) and end time (t_{end}) of the attack phase are indicated by the dashed line.

Framing For the spectral representation (see below), as well as for the calculation of the zero-crossing rate, the signal is divided into frames of 60ms length with an overlap of 50%. Since this work focusses on drum sounds, where low frequencies usually play a bigger role than in other applications, i.e. speech processing, a better resolution of those frequencies and therefore a larger window is needed. Additionally each frame is multiplied by a hamming window to reduce windowing effects.

Spectral Representation On each of the frames a fft of length $N = 4096$ is performed. For the decay spectrum only the frames containing the decay phase of the input signal are used.

2.4.2 Temporal Features

Temporal Centroid The temporal centroid can be used to describe how transient a sound is. If the sound is percussive the temporal centroid will be close to the end of the attack phase (t_{st}) and if the sound has a longer sustain it will be located more to the center of the decay phase. For the calculation of this feature the envelope $e_x(t)$ of the signal is used.

$$tc = \frac{\sum_t t \cdot e_x(t)}{\sum_t e_x(t)} \tag{3}$$

Figure 7 shows the distribution of the temporal centroid among the different classes. It

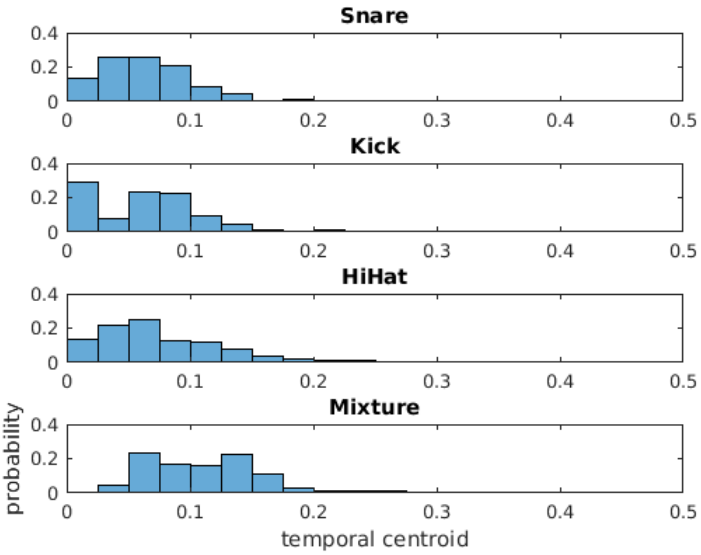

Figure 7 – Distribution of the temporal centroid, normalized to the length of the individual signal. The probability is estimated by a histogram of the training data.

can be observed that for the kick drum sounds the centroid lies at the beginning of the frame whereas more stationary sounds especially the mixtures have a temporal centroid which is more shifted to the right.

Attack Zero Crossing Rate The zero corssing rate is the number of times the waveform of the signal changes its sign relative to the length of the signal. The attack zero-crossing rate is computed as a single value over the attack phase of the signal. As can be seen in figure 8 the attack zero-crossing-rate is close to zero for sounds from the categories snare, kick and mixture and is relatively high for hi-hat sounds which are more noise-like than sounds from the other categories.

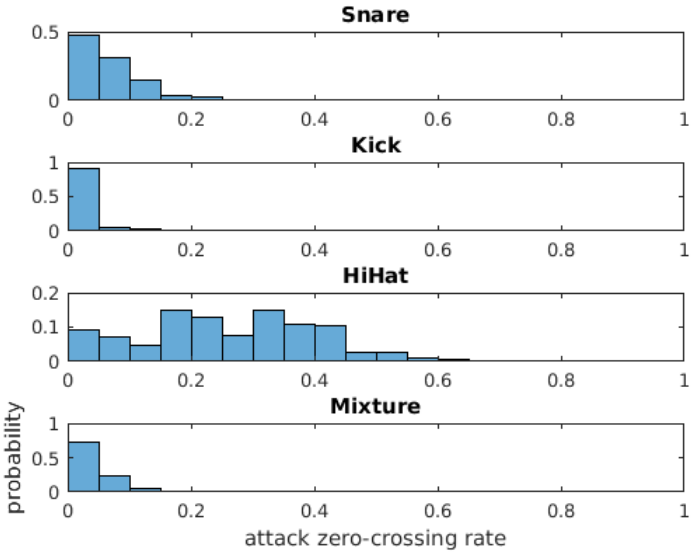

Figure 8 – Distrubtion of the attack zero-crossing rate. The value is normalized to the length of the signal.

2.4.3 Spectral Features

Decay Spectral Flatness The decay spectral flatness is defined by the ratio of the goemetrical mean of the spectrum to its mean value. For tonal signals the spectral flatness measure is close to 0, whereas for noisy signals (flat spectrum) it is expected to be close to 1.

$$dsfm = \frac{\left(\prod_{n=1}^{N} |S_d(n)|\right)^{\frac{1}{N}}}{\frac{1}{N}\sum_{n=1}^{N} |S_d(n)|} \tag{4}$$

It is computed individually for each of the frames and the mean and variance over all frames are taken as the final feature representation. Figure 9 shows the distribution of the mean and variance of the spectral flatness measure over the decay spectrum. As expected the mean values for hi-hat sounds is higher than for the other sounds since it has the most "noisy" spectrum. While the mixture sounds are expected to have the most peaky or tonal spectrum and therefore show values for the spectral flatness which are close to zero the distribution of SFM values for snare and mixture sounds show a high similarity.

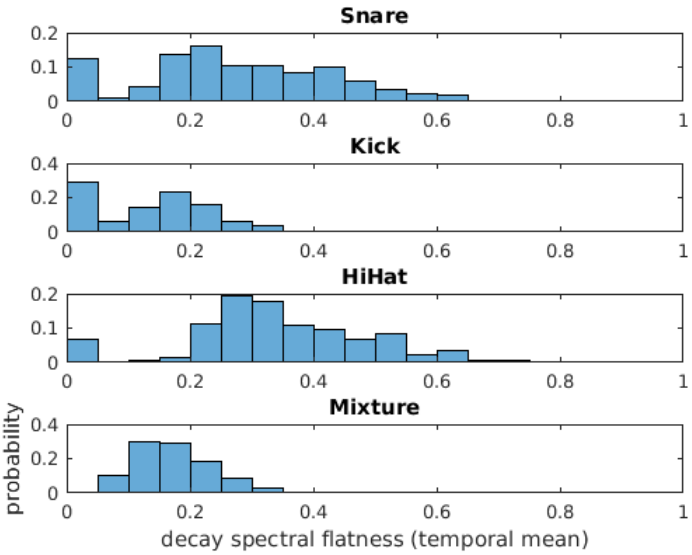

Figure 9 – Distrubtion of the decay spectral flatness measure.

Decay Spectral Centroid The decay spectral centroid is the centroid of the decay spectrum. It is computed for each frame individually and the mean over all frames is taken as the final feature representation. Figure 10 shows the distribution of the decay spectral centroid over the whole dataset. As expected for hi-hat sounds the spectral centroid is concentrated more to the higher frequencies than for snare and kick drum sounds. This can also be observed in figure 11, which shows the spectral centroid for typical snare, kick and hi-hat sounds.

$$dsc = \frac{\sum_{n=1}^{N} f \cdot |S_d(n)|}{\sum_t |S_d(n)|} \tag{5}$$

Decay Spectral Kurtosis The kurtosis of the decay spectrum is a descriptor of the shape of the spectrum. It can be interpreted as a measure for how peaked a distribution is. A high kurtosis value usually means that the distribution is more pointed, a low value on the other hand means that the distribution is more flat.

$$Kurt(S_d) = \frac{E[(|S_d(n)| - \mu)^4]}{E[(|S_d(n)| - \mu)^2]^2} \tag{6}$$

Where μ denotes the mean value of the amplitude spectrum of the decay phase. The kick drum sounds are expected to have a higher spectral kurtosis than hi-hat or snare sounds since the spectrum is less flat (see figure 12).

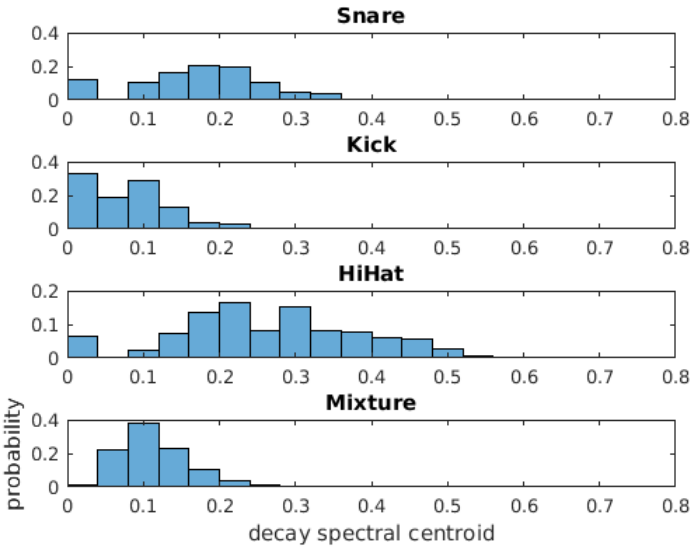

Figure 10 – Distrubtion of the decay spectral centroid (normalized frequency)

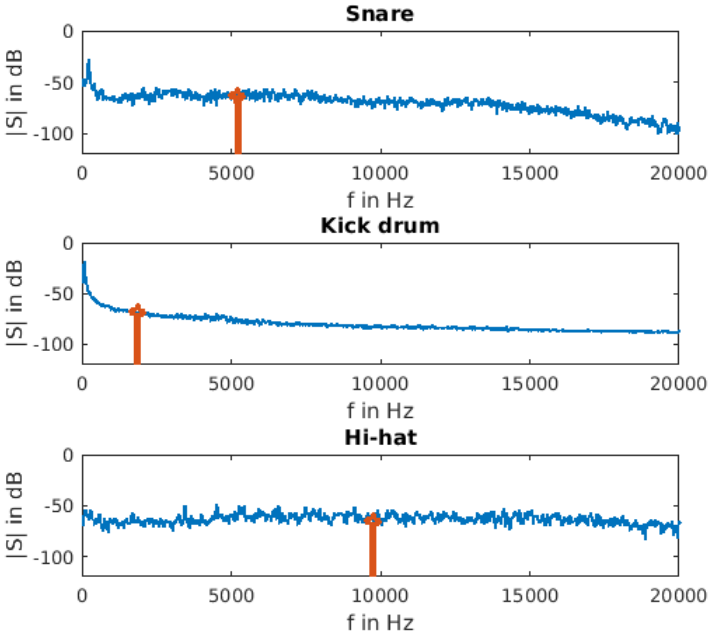

Figure 11 – Spectral Centroid of typical snare (top), kick (middle) and hi-hat (bottom) sounds.

Decay Spectral Skewness The spectral skewness is a measure for the asymmetry of the spectrum. If the skewness is negative the mass of the distribution is more concentrated to the right side. If the value is positive the right tail is longer and the mass

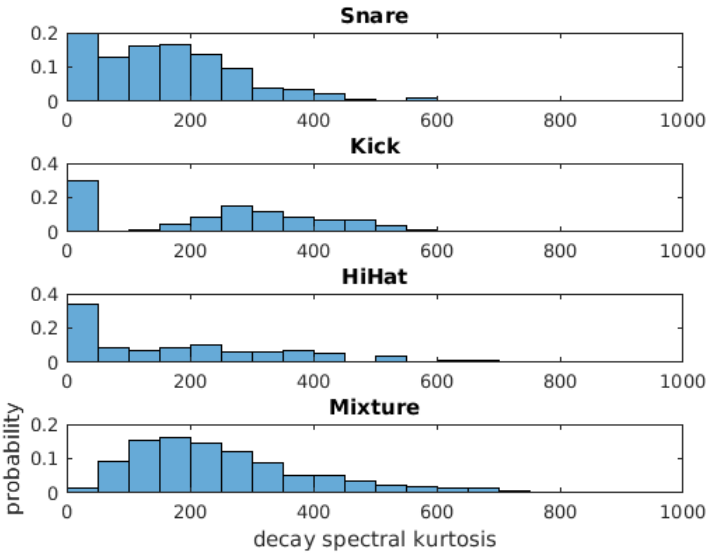

Figure 12 – Distrubtion of the decay spectral kurtosis

is concentrated to the left side of the spectrum.

$$Skew(S_d) = \frac{E[(|S_d(n)| - \mu)^3]}{E[(|S_d(n)| - \mu)^2]^{3/2}} \tag{7}$$

When looking at the distribution of the decay spectral skewness, two different peaks for the kick drum class can be observed. This may be explained by the fact that some of the sounds are essentially overlayed by hi-hat or cymbal sounds.

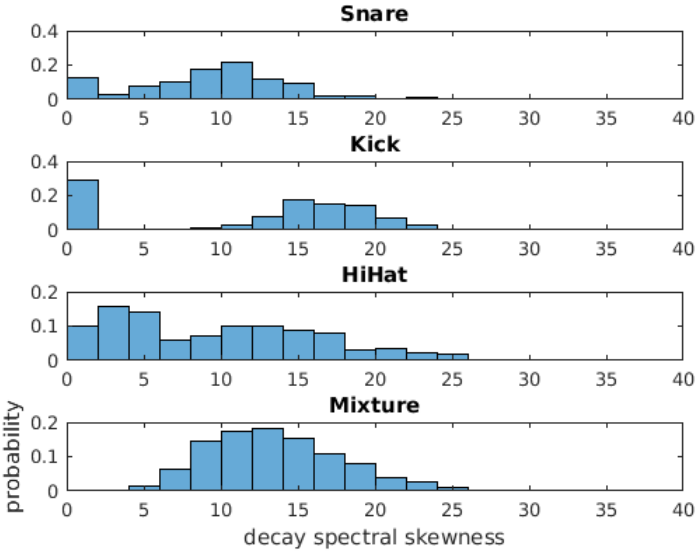

Figure 13 – Distrubtion of the decay spectral skewness

Spectral Roll-Off The spectral roll-off is the frequency point at which the spectrum has reached 85% of its overall power distribution. For this feature the logarithmic spectrum $SLog_k$ scaled to positive values is used. Here the entire spectrum, including attack and decay phase, is used.

$$SLog_k = 20\log(S_k) - \min(20\log(S_k)) \tag{8}$$

Again the mean over all frames k is used as the final value. Figure 14 shows the distribution over the whole dataset.

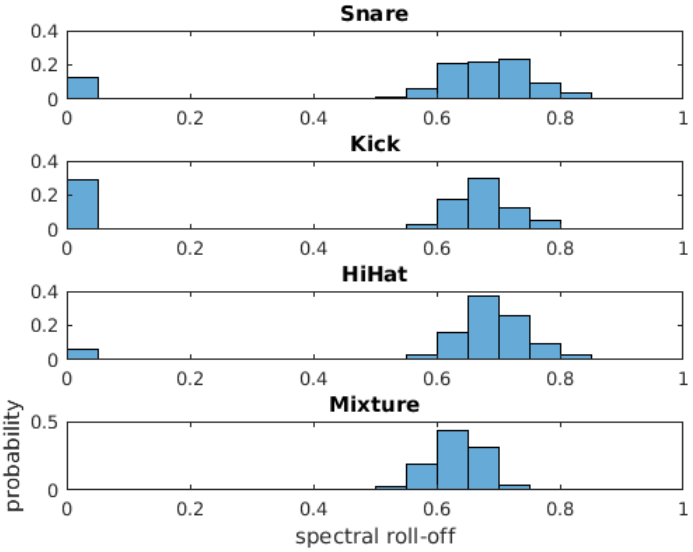

Figure 14 – Distrubtion of the spectral roll-off

MFCC's Mel Frequency Cepstral Coefficients are spectral descriptors widely used in speech processing [RJ93]. In this work MFCC's up to the 13th coefficient are calculated for each frame with the first coefficient being discarded. The means and variances over each bin are taken as descriptors, making it a total of 24 MFCC values. For the actual implementation, the code provided by Dan Ellis (Columbia University) is used [Ell05].Figure 15 shows the means of the MFCC's averaged over each of the classes in the dataset.

Bark Spectrum To represent the spectral energy distribution in a way that approximates the human hearing the bins of the spectrum are grouped into 20 bark bins. This is done by multiplying the power spectrum P by a weighting matrix $\mathbf{W_b}$. The power spectrum P is obtained by taking the average of the spectrogram S over time.

$$B = 20 \cdot \log_{10}(\mathbf{W_b} \cdot |P|) \tag{9}$$

Figure 16 shows an example of the weighting function, below 500 Hz the bark scale is linear with equal distances and above 500 Hz it is similar to a logarithmic axis. For the

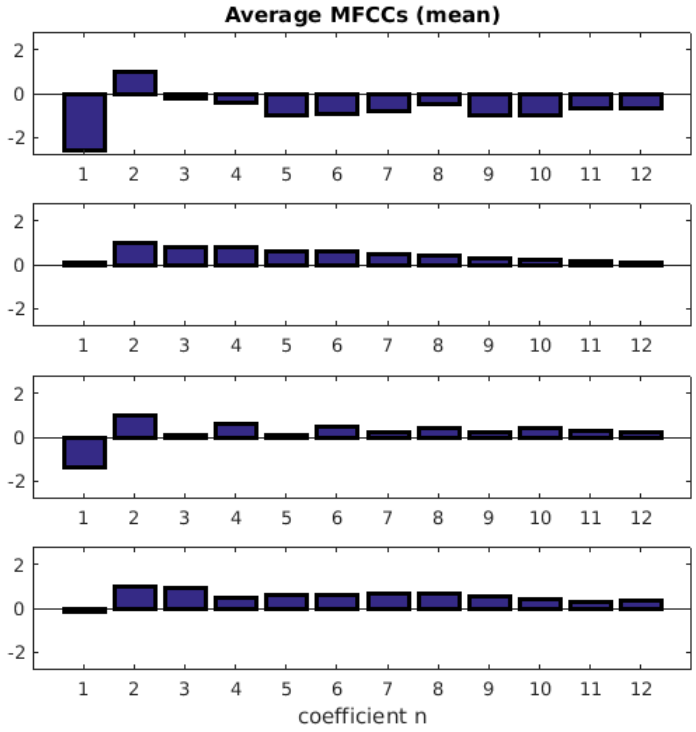

Figure 15 – MFCC's averaged over each of the classes

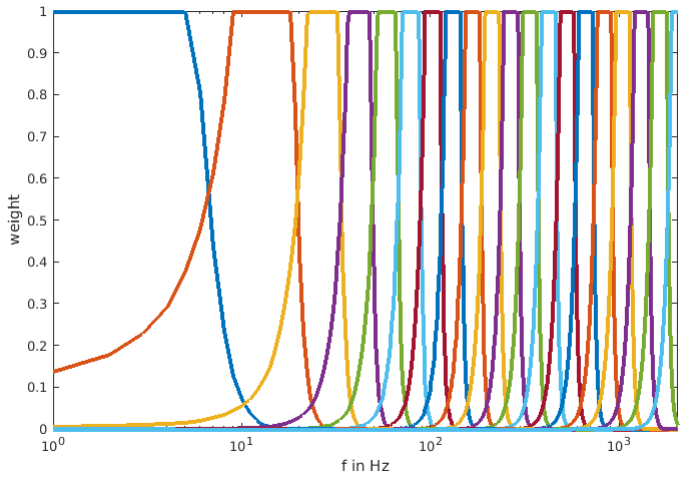

Figure 16 – Plot of the bark scale weighting matrix. Rows are shown as individual funtions.

generation of the weighting matrix we use again Dan Ellis' Rastamat toolbox [Ell05]. Figure 17 shows the bark spectra for each of the four classes averaged over the whole dataset.

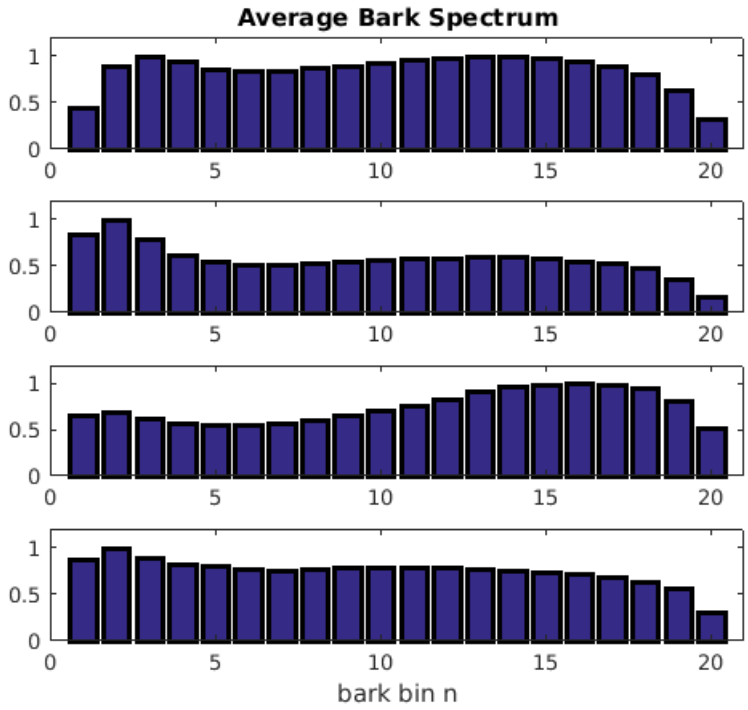

Figure 17 – Bark Band Energy averaged over each of the classes

2.5 Dataset

The dataset consists of 2698 samples. 949 of these samples are isolated drums sounds and are taken from comercial sample libraries. The other part consists of hand labeled segments from the Ultimate Breaks & Beats compilations and contains all of the four classes. The exact distribution if shown in table 1.

Class	Snare	Kickdrum	Hi-Hat	Mixture
Samples	386	309	254	0
UB&B	100	163	17	1485
Overall	486	472	271	1485

Table 1 – Dataset used for training of the classifier.

2.6 Classification

2.6.1 Feature scaling

Before any further analysis takes place the features of the dataset are standardized. This is done by calculating the standard score for each of the features x

$$x' = \frac{x - \mu}{\sigma} \tag{10}$$

where μ and σ denote the mean and standard deviation of the feature.

2.6.2 Linear Discriminant Analysis

To reduce the number of dimensions in the feature space Linear Disciminant Analysis (LDA) is used. Assuming gaussian normal distribution of the data LDA tries to maximize class seperability by analyzing the ratio of the so called between-class matrix $\mathbf{S_b}$ and the within-class matrix $\mathbf{S_w}$. For K different classes and n_k being the number of samples in this class the scatter matrices can be defined as:

$$\mathbf{S_b} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{\Sigma_k} \tag{11}$$

$$\mathbf{S_w} = \sum_{k=1}^{K} \frac{n_k}{n} (\mu_k - \mu)(\mu_k - \mu)^T \tag{12}$$

Where μ denotes the overall mean and $\mathbf{\Sigma_k}$ and μ_k the covariance and mean of the class k. Eigenanalysis is then performed on the ratio $\frac{S_b}{S_w}$ resulting in the eigenvectors Φ_i and the corresponding eigenvalues λ_i.

$$S_b S_w^{-1} \Phi_i = \lambda_i \Phi_i \tag{13}$$

The Eigenvectors corresponding to the $K-1$ largest eigenvalues are now combined to a new Matrix $\mathbf{L}$.

$$\mathbf{L} = [\Phi_1^*, \Phi_2^*, ..., \Phi_{K-1}^*]^T \text{ with: } \lambda_i^* > \lambda_{i+1}^* \tag{14}$$

This matrix $\mathbf{L}$ is of size $K-1$xM with M denoting the number of features and can now be used to transform the original dataset $\mathbf{X}$ to a K-1-dimensional space $\widehat{\mathbf{X}}$ so that the class seperability is maximized according to its mean values.

$$\widehat{\mathbf{X}} = \mathbf{L}\mathbf{X} \tag{15}$$

In the context of this work this means the dimension of the feature space is reduced from 66 to 3 dimensions. Figure 18 shows an example where LDA is applied on the previously described dataset consisting of the four classes snare, kick drum, hi-hat and mixture. Except for a few outliers the data is well seperated when looking at the first three dimensions of the transformed feature space

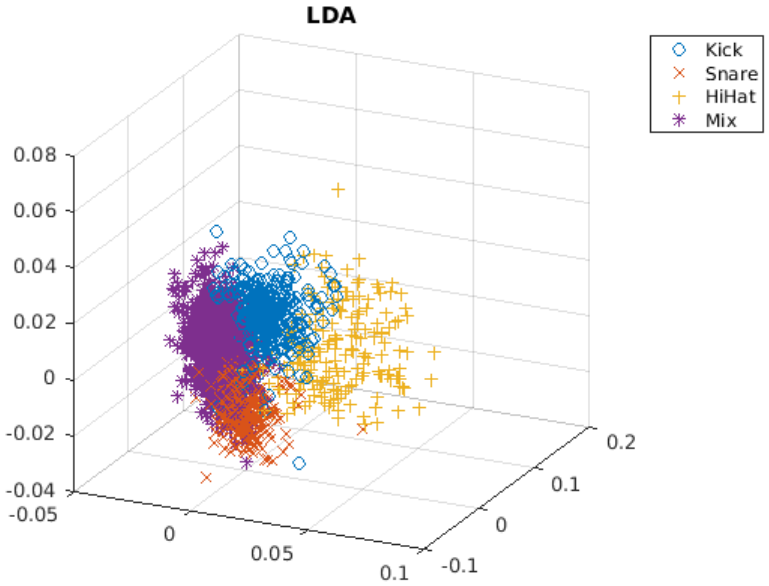

Figure 18 – LDA applied on the dataset

2.6.3 Classification: Maximum Likelihood Method

As a classifier the maximum likelihood method is used. From the transformed training data the means and variances, μ_k and Σ_k are calculated for each class k. Assuming gaussian normal distribution of the data the probabilty $p(k)$ of a new sample x belonging to class k can be calculated:

$$p(k) = \mathcal{N}(x, \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^K \det(\Sigma_k)}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)\right) \tag{16}$$

The sample x is now assigned to the class with the highest probability

$$k = \arg \max_k (p(k)) \tag{17}$$

As described before, the data is standardized by using the mean and standard deviation calculated from the training data.
It should be noted that the maximum likelihood method assumes the data to be normally distributed, which is only a rough approximation of the actual distribution (see figure 19). Of course it would also be possible to use different algorithms like k-means or gaussian mixture models for classification.

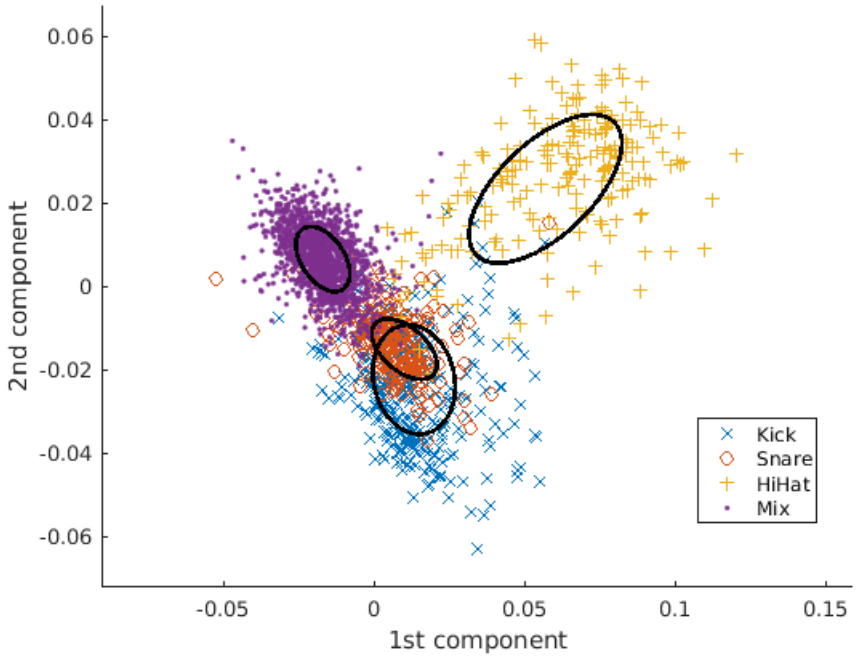

Figure 19 – The first two components of the lda. Gaussian ellipsoids are drawn for each class as a reference. It can be seen that, especially for the three classes representing the drum sounds, the assumption of normal distributed data is only a rough estimation.

3 Results

3.1 Classification Rate

The classifier is evaluated using 10-fold cross validation. That means that the data is divided into a training set consisting of 90% of the samples and a test set consisting of 10% of the data. The samples are selected randomly and then used to test the classifier. This process is repeated 10 times and the mean of the results is taken as the overall classification result. It should be noted that in particular stratified cross-validation is used here. This means each fold has roughly the same proportions as the complete dataset, regarding the distribution of the classes. As shown in table 2 an overall classification rate

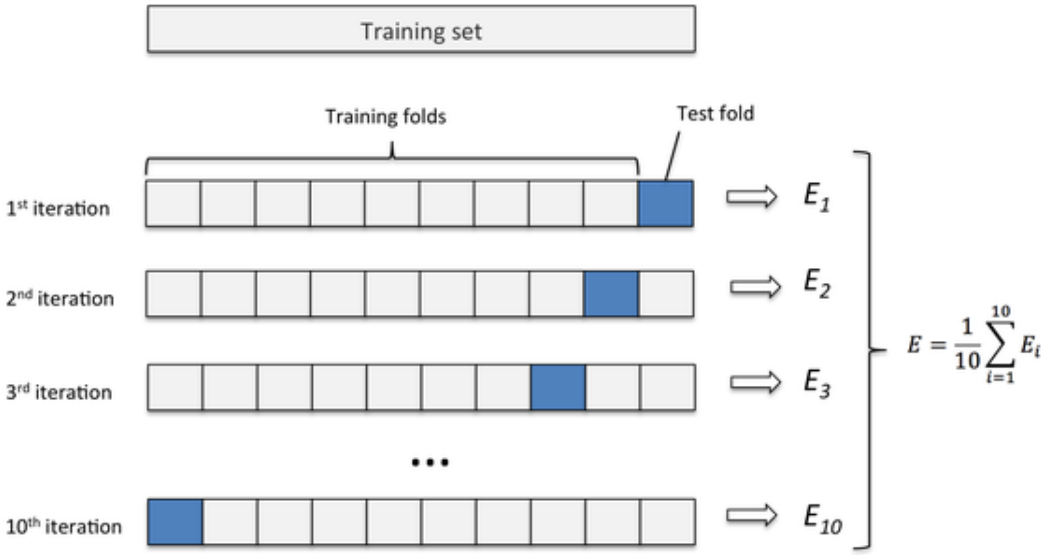

Figure 20 – Stratified 10-fold cross validation [ras]

of 89% was achieved using cross-validation on the whole dataset. Table 20 shows the confusion matrix, which was generated by adding up all the individual results from the cross-validation procedure. The highest classfication rate was achieved for the mixture sounds with 92%, which could be due to the fact that the biggest portion of the dataset represents this class (see 2.5). However the rate of 86% for hi-hat sounds is higher than that for the snare sounds which is 79%, although the dataset consists of about twice as many snare sounds than hi-hat sounds. But when looking at the distribution of the attack zcr 8, the spectral flatness 9 or the spectral centroid 10 it is not surprising that for hi-hat sounds a higher classification rate than for the rest of the drum sounds was achieved, since the hi-hats have a very distinctive spectrum and can be well seperated from the other classes.

# Iteration	Overall	S	K	H	M
1st Iteration	89.0	77.5	86.0	80.8	92.6
2nd Iteration	88.8	72.5	90.7	88.5	92.6
3rd Iteration	91.9	90.0	86.1	96.2	93.3
4th Iteration	90.7	85.0	95.4	76.9	93.3
5th Iteration	88.4	75.0	86.1	76.9	94.6
6th Iteration	91.1	85.5	95.4	88.5	92.6
7th Iteration	89.9	75.0	88.4	92.3	94.0
8th Iteration	91.1	87.5	93.0	88.5	92.0
9th Iteration	87.6	65.0	93.0	84.6	92.6
10th Iteration	90.7	85.0	86.0	80.8	95.3
Mean	89.8	79.6	90.0	85.4	93.3

Table 2 – Results of 10-fold cross-validation in percent.

	Snare(486)	Kick(472)	Hi-Hat(271)	Mixture(1485)
Snare	79.8%	13.3%	4.3%	2.8%
Kick	4.9%	90.0%	1.4%	3.7%
Hi-Hat	2.7%	10.8%	85.4%	1.2%
Mixture	1.9%	3.7%	1.1%	93.3%

Table 3 – Confusion Matrix

3.2 Testing with music

The system was tested with three different music signals of about three minutes length, each containing a short isolated drum break. The resulting hi-hat sounds were all correctly classified and no false positives were produced. However 3 out of 9 kick drum sounds and 3 out of 6 snare sounds were masked by harmonic instruments and therefore misclassified sounds. An example is shown in figures 21 and 22. When looking at the spectrum of the misclassified snare sound, a harmonic structure of multiple harmonics starting at around 140 Hz can be observed while the spectrum of the correctly classified snare sound at the bottom has a more flat spectrum with a single peak at roughly 180 Hz. Nevertheless the two spectral densities still show a high similarity and it is explicable that the current spectral features e.g. spectral flatness and spectral centroid are not sufficient to distinguish between these two kinds of spectra with high certainty.

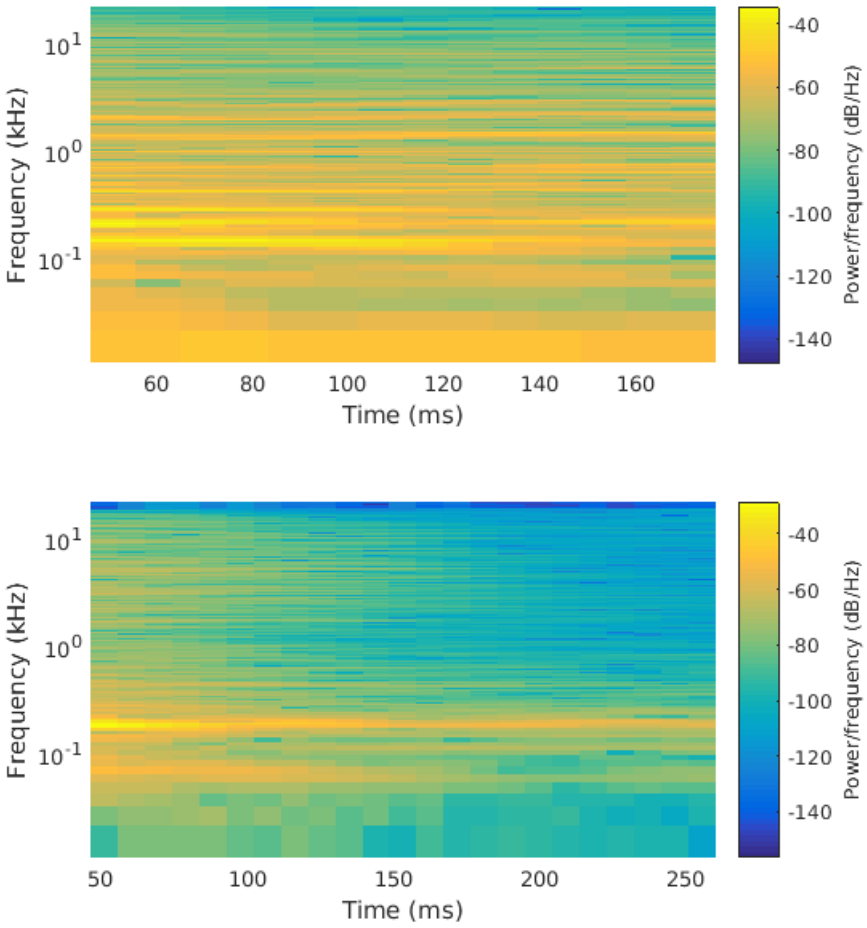

Figure 21 – Spectrograms of a misclassified snare sound masked by other instruments (top) and a correctly classified snare sound (bottom)

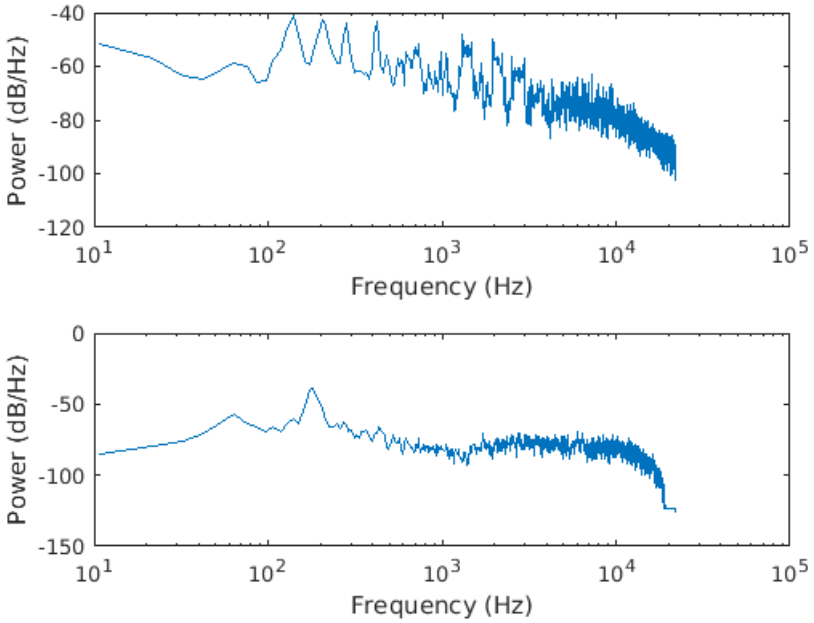

Figure 22 – PSD of a misclassified snare sound masked by other instruments (top) and a correctly classified snare sound (bottom)

4 Conclusion

In this work an algorithm to locate and classify drum sounds from digital music recordings was developed. While relatively high classification rates were achieved, the system only worked well when it was applied on recordings where isolated drum sounds are present. For recordings without any potential samples the system still produces some results that obviously can not be correct. Therefore the algorithm in its current state is not perfectly suited to analyze larger music collections since it still requires the user to manually listen through results to sort out the misclassified samples. It would be necessary to include further features which model the tonality and the harmonic structure of a sound to better seperate (harmonic) mixture sounds and drum sounds e.g. fundamental frequency and tristimulus estimation. Furthermore it could be of advantage to better represent the temporal evolution of the spectral features for example by using hidden markov models or by simply including more statistical measures. When looking at the temporal waveforms of the snaresounds in the dataset, it can be seen that the temporal evolution of the waveform can be approximated by a exponential function. Another aproach would be to use a threshold to distinguish between drum and mixture sounds. These thresholds could be defined by exponential functions by a time constant τ (see figure 23). For the use with large music collections maybe a second classification step could help to improve the classification rate in between the three drum classes. As a final remark it should be relatively simple to include more categories to the classifier for example toms, cymbals and hand claps or a differentiation between open and closed hi-hats by including more samples to the training set.

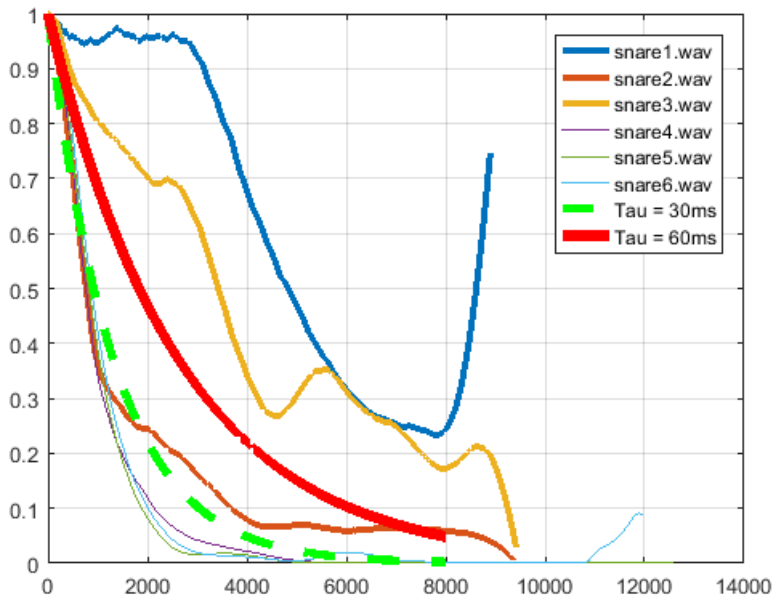

Figure 23 – Temporal patterns of examplary snare sounds and threshold functions. The temporal curve is lowpass filtered and normalized to the maximum value. `snare1.wav`, `snare2.wav` and `snare3.wav` are wrongly classified snare sounds masked by other (harmonic) instruments. The thick red ($\tau = 30$) and green ($\tau = 60$) lines represent exponential functions for the corresponding time constant τ.

References

[BKS] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods."

[Ell05] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005, online web resource. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

[Ell07] ——, "Beat tracking by dynamic programming," *J. New Music Research*, vol. 2007, pp. 51–60, 2007.

[Ess] "Sample. ein verwaschener begriff, erklärt von karlheinz essl," http://www.essl.at/bibliogr/musiklexikon-sample.html, accessed: 2017-02-10.

[HYYG02] P. Herrera, A. Yeterian, R. Yeterian, and F. Gouyon, "Automatic classification of drum sounds: A comparison of feature selection and classification techniques," in *Proceedings of 2nd International Conference on Music and Artificial Intelligence*. Springer, 2002, pp. 69–80.

[Pee04] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Icram, Tech. Rep., 2004.

[ras] "Machine learning faq, seabstian raschka," https://sebastianraschka.com/faq/docs/evaluate-a-model.html, accessed: 2017-02-10.

[RJ93] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[ZPDG02] A. Zils, F. Pachet, O. Delerue, and F. Gouyon, "Automatic extraction of drum tracks from polyphonic music signals," in *Second International Conference on Web Delivering of Music, 2002. WEDELMUSIC 2002. Proceedings.*, 2002, pp. 179–183.

5 Appendix: Documentation of the code

5.1 Feature Extraction: computeFeatures.m

Computes audio features for given signal x
[Data, Labels] = computeFeatures(x,fs) calculates all audio features and saves them as a single vector in Data, Labels is a cell array containing the description of each value in Data

[Data, Labels] = computeFeatures(x, fs, FEATURE) calculates the feature(s) specified in FEATURE and saves them as a single vector in Data, Labels is a cell array containing the description of each value in Data

Features include:
'temp-centroid'
'attack-zcr' attack zero-crossing-rate
'dec-sp-flatness'
'dec-sp-centroid'
'dec-sp-kurtosis'
'dec-sp-skewness'
'dec-zcr' decay zero-crossing-rate
'spec-bands' relative bark band energy
'mfcc' Means and Variances of the mfcc's 1-13
'corr' Correlation coefficients 1-12

The features calulated when calling the function without the FEATURE argument, are specified by the variable varargin.
Executing the function with empty arguments [, L] = computeFeatures([],[]), returns the labels of the features as strings inside the vector L. For exmaple: L{3} returns the name of the third feature.

The function getFeatures.m calls computeFeatures for any given filename. For example: [data, labels] = getFeatures('snare1.wav').

Files needed by computeFeatures.m :
- files contained by /features
- files contained by /rastamat

5.2 Generation of the training data: get_data.m

In this script the features for the training data are computed and saved to the file 'train_data_kshm.mat' together with the labels and the mean and variance values, necessary for feature standardization.

After that linear discrimant analysis is performed on the data. The resulting transformation matrix and the statistics of the transformed data are saved as a structured variable to the file 'lda_data_kshm.mat'.

Files needed by `get_data.m` :
- `computeFeatures.m`
- `getFeatures.m`
- `standardize.m`
- `ldaDrums_Mix.m`
- files contained by `/features`
- files contained by `/rastamat`

5.3 Cross validation: `evaluate_model.m`

In this script 10-fold cross validation is performed using the training data. After partitioning the data, training and testing is performed for all the combinations. For classification the maximum likelihood method is used.
The probabilities are computed using the function `mvnpdf(testData, mean, covariance)`

Files needed by `evaluate_model.m` :
- `train_data_kshm.mat`
- `partitionVector`
- `standardize.m`
- `ldaDrums_Mix.m`

5.4 Main program: `prototype.m`

Here the alforithm can be tested using a audio file of choice. Running this script will write the extracted sounds to cell arrays: `SnareCandidates`, `KickCandidates` and `HiHatCandidates`. For each categoruy the `N_best` sounds with the highest probabilty are saved to the cell arrays: `BestSnares`, `BestKicks` and `BestHiHats` and can be saved to a specified folder. Note: The code for saving the sounds at the end of the script is commented out by default.

Files needed by `prototype.m` :
- `train_data_kshm.mat`
- `lda_data_kshm.mat`
- `crestFactor.m`
- `silenceRatio.m`
- `computeFeatures.m`
- `getFeatures.m`

- `partitionVector`
- `standardize.m`
- `ldaDrums_Mix.m`
- files contained by `/features`
- files contained by `/rastamat`
- files contained by `/onset_detection`