AUTOMATED TRANSCRIPTION OF DRUMSOUNDS

Anyere Bendrien, Maximilian Wagenbach

Technical University of Berlin
Audiocommunication Group
Einsteinufer 17, 10587 Berlin

ABSTRACT

While humans are naturally capable of detecting patterns like percussive events in music such a task is difficult for a computer. In this research we analyze the possibilities of finding the time points of drum events in a piece of music by using non-negative matrix factorization. Generating a so called drum transcription can have lots of different applications. By using well established techniques based on preceding research we hope to get a reliable drum transcription algorithm with detection rates of up to 90%.

***Index Terms*—** Drum Transcription, Non-Negative Matrix Factorization, Onset Detection, Music Information Retrieval

1. INTRODUCTION

For this research we look into the topic of generating a transcription of drums from a piece of music containing a recording of a drum kit. Drum transcription is considered a classical music information retrieval (MIR) problem, where you want to extract information from a mixed piece of music. In order to achieve this an algorithm is applied to the raw sound data to find the time points where a drum event happens in the song. This list of time points and a corresponding flag with the kind of the drum event is called the drum transcription. While in theory this can be used to find any kind of event within a musical piece, we focus mainly on basic drum sounds in our research.

The transcription of the drum track of a song can be useful in many cases. The simplest would be to generate the sheet music for a song where the original has been lost or is not available. It can also be used for more sophisticated use cases like a conversion of a audio file to a MIDI sequence (known as Audio-to-MIDI). For syncing visuals to music in VJing or light show applications (Sound-to-Light) a drum transcription is also an integral part. Furthermore the drum transcription of a song can be used as a basis for more complicated MIR analysis. For example using the results of the drum transcription a Beats per Minute detection can be build, which in turn can then be the basis of a tempo detection algorithm. Overall the drum transcription of a song is a basic but versatile information, that can be used in many different applications.

On first sight extracting the time point of a drum hit for a polyphonic, mixed piece of music sounds like a difficult task. In fact though there are multiple algorithms to achieve that. For this research we take an algorithm commonly used for these kind of problems: the non-negative matrix factorization (NMF).

After introducing the current state in research of this subject area in section 2 we give a detailed explanation of how the NMF and our algorithm works in section 3. In section 4 we evaluate our results and come to a conclusion in section 5.

2. RELATED WORK

The idea to use NMF for pattern recognition was first proposed by Lee and Seung in 1999. [1] As described in their paper NMF is a versatile technique to find features in a data set. Lee and Seung proposed in their paper that it can be used to recognize parts in faces, find semantic features in texts or find certain musical events in a piece of music. During the last decade NMF was researched and applied in all the fields they mentioned and many more.

In the field of MIR NMF has also been used in a variety of applications. Most notable are the fields of source separation and music transcription. For source separation the goal is to separate a piece of music containing a mix of multiple sound sources into the their components. [2]

In the field of music transcription NMF has also been used for several topics. Smaragdis and Brown [3] used NMF to analyze and extract polyphonic, harmonic content from a piece of music. In the sub-field of drum transcription there has also been some research in the recent years. Paulus and Virtanen [4] as well as Moreau and Flexer [5] as well as Wu and Lerch [6] have all shown that NMF is a technique that can be well applied to the problem of finding the time points of percussive events in music.

Thanks to Holger Kirchhof and Tim Flohrer for assistance. Also thanks to TELECOM ParisTech - Dep. TSIe for generously providing the ENST drum test data set.

3. ALGORITHM OVERVIEW AND DESCRIPTION

As mentioned above we use a **non-negative matrix factorization** to extract the time points of drum events from a drum track. In general terms the NMF factorizes a matrix (usually called $\mathbf{V}$) into two smaller matrices (usually called $\mathbf{W}$ and $\mathbf{H}$) with the constraint that every matrix element must be non-negative. In turn the sum of $\mathbf{W}$ and $\mathbf{H}$ is an approximation of $\mathbf{V}$ (see figure 1).

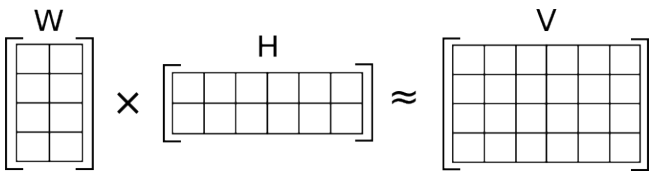

Fig. 1. Matrix $\mathbf{V}$ can be factorized into two matrices $\mathbf{W}$ and $\mathbf{H}$, so that their sum approximates $\mathbf{V}$. © WikiMedia CC BY-SA 3.0

In our case we use the magnitude spectrum of a song as the $\mathbf{V}$ matrix and decomposed it into two matrices. The first one contains the magnitude spectra of the individual components we are interested in ($\mathbf{W}$) and the second one contains the time-varying gain of these components ($\mathbf{H}$), which tells us when each of the components for the first matrix is active. For that reason the $\mathbf{W}$ matrix is usually called component matrix and $\mathbf{H}$ is usually called activation matrix. Again the sum of both of these matrices makes up an approximation of the original spectrum [3]. A visual representation of all these matrices can be seen in figure 2.

For our use case we are not interested in finding all the components and their activations. We are rather interested in finding the corresponding activation of a specific component, like that of a base drum or a hi-hat. This is also possible using NMF. In order to do so, we specify $\mathbf{W}$ as the spectrum of a base drum (see figure 2 (b)) and $\mathbf{V}$ continues to be the spectrum of the whole song (see figure 2 (a)). The NMF then solves for the matrix $\mathbf{H}$ (see figure 2 (c)) which tells us at what frame the specified base drum spectrum appears in the song. In that setup the NMF essentially works as a pattern matching algorithm.

Figure 3 shows a flowchart with the complete overview of the algorithm we developed. On the left hand side the two inputs to the algorithms are shown. The first input is a time domain signal of percussive instrument, like a base drum, and the second input is a time domain signal of a piece of music. For both inputs the first step is to down mix the signals into a mono signal. Next the short-time Fourier transform (STFT) is applied, which transforms both signals from the time domain to the frequency domain. For the parameters of the STFT we use a frame size of 1024 samples which yields in a frequency resolution of approx. 43 Hz per bin. We also chose an overlap of 50% between hops to improve the time resolution. A visual representations of both transformed inputs can be seen

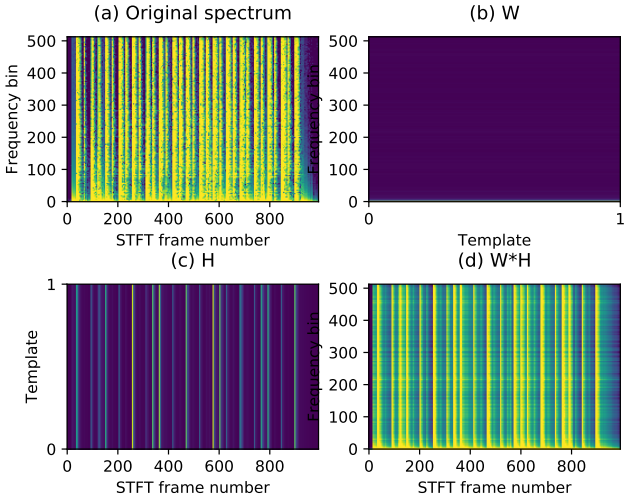

Fig. 2. (a) Magnitude spectrum of a drum track. (b) Magnitude spectrum of a base drum. (c) Activation matrix of the base drum. (d) Approximated spectrum. © Own work CC BY-SA 3.0

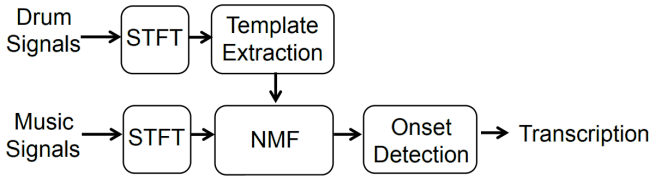

Fig. 3. Flowchart of the whole drum transcription algorithm. © Own work CC BY-SA 3.0

in figure 2 (a) and (b) respectively.

Before both inputs are fed into the actual detection the drum sound undergoes some further processing. This step is marked as **template extraction** in figure 3. For our base drum template we chose to only use the transient of the base drum. This has the advantage that it makes the template independent of the length of the original base drum sound. As most of the energy is in the transient it is the most relevant part of the base drum sound. Using only the transient also makes the template somewhat independent of the room it was recorded in, as any possible reverberation or echo is cut off.

After both inputs are prepared they are handed over to the NMF, which does its processing in the way described before. The result of the NMF is in our case the activation matrix $\mathbf{H}$. The matrix $\mathbf{H}$ can also be viewed as a function curve of how much the given template is contributing to the given music signal over time. The next step of the algorithm is called **onset detection** where we extract the time points for an percussive event from the matrix $\mathbf{H}$. An alternative representation of the matrix $\mathbf{H}$ as a curve can be seen in figure 4. The onset detection is a simple implementation loosely based on the algorithms described by Lerch [7] in chapter 6.3. To extract the onsets of the drum events the significant local maxima of

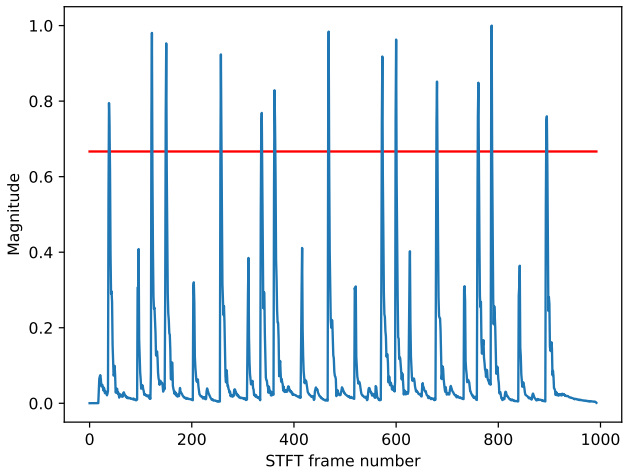

Fig. 4. Example of the representation of an activation matrix as a curve. With a red line marking the threshold at $\frac{2}{3}$. © Own work CC BY-SA 3.0

the curve (peaks) have to be detected. In order to do that the curve is first normalized. Afterwards a threshold of $\frac{2}{3}$ is applied, which can be seen as a red line in figure 4. This value was establish empirically to separate the valid detections from false positives. Then the first derivative, which represents the slope of the curve, is calculated, by subtracting each element from the previous one. An example of the resulting curve can be found in figure 5. In order to find the extract time points of a peak the zero crossings of the resulting curve have to be detected. Once the frame number of a zero crossing is found, the frame number has to be converted back to a time point. This is achieved by applying the following formula, where f_s is the sample rate.

$$timePoint = zeroCrossingFrameNumber * \frac{hopSize}{f_s}$$

Due to this formula we are only able to get the time point within a range of approx. 12 ms (called epsilon). Once this calculation is done we receive a list of time points where the given template is found in the song which is the drum transcription we were looking for.

The implementation of our algorithm is done in Python using the numpy/scipy library stack. As well as the library nimfa[1] for NMF calculations. The entire code repository is licensed under the GPL 3.0 and can be found on GitHub[2].

[1]NMF library: `http://nimfa.biolab.si/`

[2]Repository: `https://github.com/Bendrien/AudioContentAnalysis`

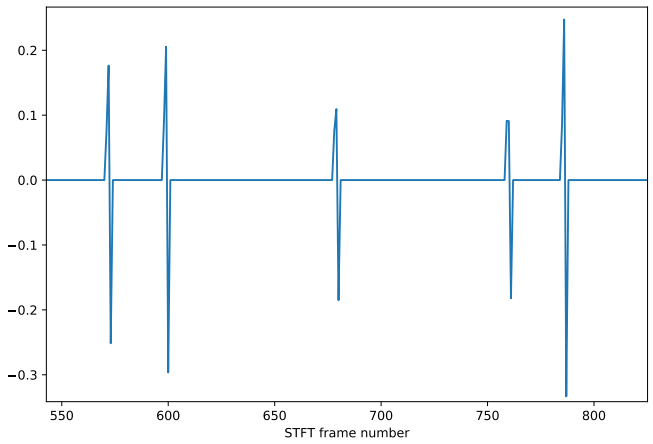

Fig. 5. An example of the first derivative of the activation function. © Own work CC BY-SA 3.0

4. EVALUATION

As our test data set we used the ENST-drums database of the Institut Télécom Paris Tech[3]. The data set contains recordings of 207 drum tracks split over 3 different drum kits. In the data set each instrument of the drum kit is recorded separately. For our purposes we use the mixed version of the tracks called "wet-mix" which also contains some acoustic elements of the recording room, like reverberation, making it more realistic for real world applications. Next to the recordings the test data set also contains an annotation file for each drum track, which lists the occurrence of an instrument as well as the time point it occurred. This data was used as the ground truth to compare our algorithm against.

In order to evaluate the accuracy of our algorithm we compared the drum events that our algorithm detected with the annotations of the ENST test set. When comparing the results we took into account the epsilon error margin of approx. 12 ms. This leaves us with a detection rate for each track. To get an overall estimate of the accuracy of our algorithm we calculated the median for each drum kit as well as the mean and the standard deviation. Besides the true positives we also counted the false positives for each track, where our algorithm would detect a drum event, but according to the annotations there is none there. We calculated the median, mean and standard deviation across all tracks for the false positives as well. The results of our evaluation for the base drum template can be found in table 1.

As visible in table 1 the results for drum kit 2 are the best with a average of around 90% accuracy and a fairly low standard deviation. The same is true for drum kit 3 which had a detection rate in the mid 80's. Both had none or close to no false positives. Drum kit 1 shows different accuracy results. Here only around 25% of the base drum hits are detected cor-

[3]ENST drum test database: `https://perso.telecom-paristech.fr/~grichard/ENST-drums/`

Drum kit	1	2	3
# Files	63	75	69
Median$_P$	0.25	0.93	0.86
Mean$_P$	0.27	0.89	0.80
σ_P	0.22	0.12	0.18
Median$_{FP}$	0.21	0.0	0.05
Mean$_{FP}$	0.34	0.02	0.07
σ_{FP}	0.31	0.04	0.07

Table 1. Results of the base drum detection. $_P$ represents the true positives and $_{FP}$ the false positives.

rectly, while the false positive detection is also at around 21%. The standard deviation is also significantly higher.

From these results we can see that our algorithm returns very satisfying results for 2 out 3 drum kits from our test data set. We also found an explanation for the outlier. When listening to the base drum sounds of the individual drum kits it becomes obvious that drum kit 2 and 3 sound very alike with a sharp, crisp base drum sound, while the drum kit 1 sounds more smooth and mellow. Since the base drum template used here was extracted from the drum kit 2 it is not very surprising that the algorithm can't find it in the much different sounding drum kit 1. The solution to this problem would be to use a more generalized base drum template. This shows an apparent flaw in the design of the algorithm. It can only work as good as the templates it is provided with. If the sound the algorithm is looking for is pitched or distorted to much, it won't be picked up. More on how to improve this work in section 5.

5. CONCLUSION

The algorithm returns good results given its relative simplicity. As with all pattern matching algorithms if the feature the algorithm is looking for differs to much from the provided template the accuracy declines, even if a human listener is still able to pick them out. Work by Lerch et al. [8] has shown that by introducing an additional step into the algorithm this problem can be mitigated. The step is called template adaption and happens between the NMF and the onset detection in figure 3. Through different methods described in their paper it is possible to iteratively adapt the template to the detections in the music. That way an "over-fitting" of the template to a certain drum kit can be avoided.

Another area of improvement is the onset detection with special focus on the peak picking algorithm. Here the empirically chosen threshold value of $\frac{2}{3}$ already yield good results, but it is also responsible for some missed events and some false positives. Using an adaptive threshold would further improve the outcome.

As mentioned above the precision of the detected peak time point is only precise within an epsilon range of approx. 12 ms since only the STFT frame it occurring in can be tracked. By interpolating between nearby frames the accuracy of the point of inflection could be improved making the epsilon range significant smaller.

Originally we wanted to test our drum transcription algorithm on a piece of mixed polyphonic music instead of just drum tracks. Sadly we were unable to find an annotated test data set that fit our needs as a ground truth. Theoretically by using mixed music only the activation matrix $\mathbf{H}$ should get a bit more noisy. Everything else should work as described here.

In closing we have presented an algorithm that gives satisfying results when trying to exact the time points at which base drum hits happen in a drum track. It can easily be extended to other percussive instruments and maybe even other melodic instruments. The accuracy and precision could be improved in future work by the methods mentioned.

6. REFERENCES

[1] Daniel D Lee and H Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788, 1999.

[2] Tuomas Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on audio, speech, and language processing*, vol. 15, no. 3, pp. 1066, March 2007.

[3] Paris Smaragdis and Judith C Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.* IEEE, 2003, pp. 177–180.

[4] Jouni Paulus and Tuomas Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *Signal Processing Conference, 2005 13th European*. IEEE, 2005, pp. 1–4.

[5] Arnaud Moreau and Arthur Flexer, "Drum transcription in polyphonic music using non-negative matrix factorisation.," in *ISMIR*, 2007, pp. 353–354.

[6] Chih-Wei Wu and Alexander Lerch, "Drum transcription using partially fixed non-negative matrix factorization," in *Signal Processing Conference (EUSIPCO), 2015 23rd European*. IEEE, 2015, pp. 1281–1285.

[7] Alexander Lerch, *An introduction to audio content analysis: Applications in signal processing and music informatics*, John Wiley & Sons, 2012.

[8] Chih-Wei Wu and Alexander Lerch, "Drum transcription using partially fixed non-negative matrix factorization with template adaptation.," in *ISMIR*, 2015, pp. 257–263.