Analysis and Visualisation of Music

Michael Taenzer
Semantic Music Technology Group
Fraunhofer IDMT
Ilmenau, Germany
michael.taenzer@idmt.fraunhofer.de

Burkhard C. Wünsche
Department of Computer Science
University of Auckland
Auckland, New Zealand
burkhard@cs.auckland.ac.nz

Stefan Müller
Institut für Computervisualistik
Universität Koblenz-Landau
Koblenz, Germany
stefanm@uni-koblenz.de

***Abstract*—Music is characterised by the ensemble of rhythm, harmony and melody. There are many meaningful ways to transcribe these parts into human-readable formats. Typically, they require a certain degree of knowledge in music for interpretation. For people who would like to understand more of music, but lack the necessary time to gain more knowledge, these formats are unsuited. In this work, we aspire to create a more direct connection between the listener and the parts that make up music by visualising music. We create a system capable of decomposing music into its parts and then use lights which follow the singing voices and melodic instruments of a song. For this matter, we conduct a study into source separation methods to derive possible future applications in automated (unsupervised) music visualisation. Our goal is achieved using algorithms based on non-negative matrix factorisation for the separation of sources and a pitch detector to estimate fundamental frequencies at predetermined time intervals. For each isolated instrument, the frequency data, in relation to these time intervals, then controls our visualisation in sync with the audio. We present a prototype showcasing the general idea, and evaluate to what extent it fulfills our vision of enhancing the music listening experience. The results indicate that the system is generally successful and has great potential, but needs optimisation for a more unsupervised context.**

***Index Terms*—MIR, NMF, music visualisation, audio-visual systems, visual communication, source separation, pitch detection**

I. Introduction

Many reproducers for audio signals (stereo systems, smartphones, software on computers) offer visualisations, which can be displayed in real-time along with a simultaneously playing audio file. The visualisations can range from simple frequency spectra with a few frequency bands to relatively elaborate animations with a colourful mix of abstract forms reacting to volume, rhythm and frequency spectrum (Milkdrop[1], Plane9[2], Resolume[3]). Alas, just by looking at those visuals, it is near impossible to find concrete representations of single instruments and observe how they behave as the music continues playing. Therefore, they mostly act as a playful gimmick, yet - together with handcrafted animations - they are already impressive enough to occasionally feed entertainment shows in respect to abstract art or moody color and light installations[4]. This shows the potential in pushing these visualisations away from their niche into a greater focus - especially if they are able to convey more information, such as the presentation of a singer's or an instrument's melody the moment it is played within the song.

Our goal is to assert whether visualisation of melodic components can enhance the music listening experience in a way such that a listener without any musical knowledge can correlate what is seen with what is heard and thereby facilitate understanding how the music is built. This leads to many tasks and small goals in-between, which are all related to the research field of Music Information Retrieval (MIR): In order to obtain information deeply engrained inside the audio - e.g., different sources - and pave the way to present it, much more extensive audio analysis is needed than a simple transformation of the signal into the frequency domain. Such analysis quickly becomes incapable of real-time.

In terms of actual visualisation, we envision lights with trails in different colours, one for each melodic instrument in a song. To clarify its connection to an instrument, the light moves up and down in accordance to the melody the instrument plays. This is possible by evaluating fundamental frequencies and translating them into movement locations on a 2D plane, for example. The intensity or size of a light can be controlled by the volume of a played note. All in all, the lights will seem to "draw" the melody while the music plays. This idea closely resembles piano roll displays inside digital audio workstations [1]. We believe that this could enable or enhance the way of perceiving and understanding music, as listeners with musically untrained ears can also see what is happening. With all necessary data obtained, virtually any kind of visualisation is possible.

II. Previous Research and Applications

Several previous studies investigated the design of light installations, mapping of musical elements to visuals in terms of color and forms, and generally creating abstract shapes to visually enhance the listening experience using amplitude, rhythmic, and harmonic features [2], [3].

One of the earliest hardware devices providing a visualisation of live audio signals is the Atari Video Music[5] from 1977, offering a few simple animations. Typically, lighting

[1]http://www.geisswerks.com/milkdrop/

[2]https://www.plane9.com/

[3]https://resolume.com/

[4]E.g. the laser shows at the Auckland Stardome: https://www.stardome.org.nz/show/music-laser-lights-rock-the-dome-dark-side/

[5]http://www.atarihq.com/dedicated/videomusic.php

technicians utilise authoring software to create elaborate light shows for live stage performances. More recently, bands and other artists make use of Xylobands[6] during concerts and performances in order to engage with their audience during certain portions of songs and extend the feeling of immersion.

Interesting relevant recent research includes the visualisation of music using 3D cymatics, a type of modal vibrational phenomena often seen in water [4]. O'Hanlon et al. propose a system for automatic music transcription by deriving a pitch-time representation through decomposition of a spectrogram with a dictionary of pitch-labelled atoms [5].

III. Assumptions and Goals

We make a few assumptions to minimise technical requirements. For the movement of lights, we want to be able to properly distinguish between notes, not to recreate a pitch-perfect environment. Considering source separation and individual audio streams of isolated instruments, we therefore aim for a degree of separation such that the desired instrument is salient and interferences/residues from others are attenuated. In conjunction with a threshold parameter, this should meet the requirements of a pitch detection algorithm in order to correctly approximate fundamental frequencies of the salient instrument. We can then retrieve the desired fundamental frequencies. This enables us to synchronise the results during playback of the original audio file and obtain a real-time visualisation.

IV. Proposed Method

A. Technical Overview

a) Non-negative Matrix Factorisation: We investigated and evaluated different algorithms based on non-negative matrix factorisation (NMF) and cost functions. NMF follows the idea that perception of a whole is based on perception of its (additive) parts. The technique detects latent information in input data in the form of patterns and their temporal distribution [6]. This suits our core idea very well as we want to isolate (subtract) instruments from a whole. Previous research demonstrated NMF as an effective and reliable technique for source separation [7].

NMF decomposes a matrix Y with dimension $m \times n$ into two output matrices A and X with dimensions $m \times k$ and $k \times n$, respectively. Matrix A holds the patterns and matrix X the coefficients, which represent temporal activations of the patterns. The product V of these two matrices is an approximation of the original Y, typically leading to a compression or dimensionality reduction[7]:

$$Y \approx V = A * X \tag{1}$$

The goal is to minimise the error between Y and the V. A critical variable for this is the dimension k for A_{cols} and X_{rows}. The parameter k is commonly called *(number of) components*. It largely depends on the kind of input data, its length and complexity. The original authors suggest $(m+n)k < mn$ [6]. For our empirical experiments with simple and short audio excerpts of up to 10 seconds, we choose k in the range of $[10, 100]$.

Sparse NMF (SNMF) is based on the assumption that detected components always appear with a certain amount of sparseness. Pieces of music for example can be viewed as a sparse representation of their elements in a way such that all of the instruments and notes never appear at once, but rather in an alternating way. SNMF puts sparsity on the components as a user-defined constraint and thus aims to overall improve spectral properties and temporal activations in cases where "the sparseness achieved by NMF is not enough" [8], [9].

We evaluate the performance for both NMF algorithms for our application using the following metrics / cost functions:

- Euclidean distance (ED)
- Kullback-Leibler divergence (KD)
- Itakura-Saito divergence (IS) [10]

b) Fourier Transformation: One evident way to obtain Y is by extracting spectral features of an audio file by executing a series of Short-Time Fourier Transformations (STFT). This is done by hopping along an audio signal, slicing it into short segments (windowing) and applying a window function to cater for spectral leakage, before calculating the individual frequency spectra. The result is a spectrogram, a time-frequency-matrix containing magnitude intensity information (phase information is discarded). The standard parameters used in our work for audio files with a sample rate of $f_s = 44.1$ kHz are:

- Window size N: 4096
- Hop size: 512
- Window function $w(n)$: Hamming

This leads to an overlap of 87.5% and a time resolution of 0.012s. With f_s/N we yield a frequency resolution of approx. 11 Hz per frequency bin, giving us a fairly reasonable resolution, e.g. for bass lines, which play in the lower frequency bins, as well.

c) Resynthesis: Because spectrograms discard phase information, we make use of the LSEE-MSTFTM resynthesis algorithm (least squares error estimation on modified STFT magnitude) [11]. It allows us to sonify any given spectrogram by estimating the necessary phase information. In an iterative process, the magnitudes from an arbitrary signal are substituted by those from the given spectrogram V. This essentially finds a new (audible) signal close to the one in V. We initialise the algorithm with a signal made up of zero-samples and original phase information obtained by STFT (which lead to Y). For our datasets (see below), we end this process after 50 iterations. Early stopping is implemented for the case that phase accuracy improvements become negligible ($< 0.1\%$).

d) Fundamental Frequency and Pitch Estimation: A tone played by a classical music instrument such as the piano has certain timbral aspects, consisting of a number of frequencies called harmonic and inharmonic overtones. The actual frequency, which is mainly correlated in the identification of the

[6]http://xylobands.com/

[7]We will use these names of the matrices throughout this document.

pitch of the tone, is the fundamental frequency f_0. Typically, this is the one which is perceived by the human ear.

A pitch estimation algorithm has the task to find f_0 in a given audio window. We choose the YIN algorithm [12] as it stands out in the field of MIR and speech recognition with real-time applicability (low latency) and low error rates. It also alleviates the problem of f_0 often not being the most prevalent frequency or even missing, which can lead to "octave jumps" in the estimation. In our context, we feed YIN with the resynthesised audio files of the isolated instruments. We keep its threshold parameter at 0.1 as suggested by the original authors.

e) Synchronisation: Our YIN implementation generates a CSV file with pitch values in given time segments/frames. In our experiments, the frame size is equal to the window size used for STFT. During audio playback, we retrieve the current time position from the sample buffer and read the corresponding pitch slice containing the desired f_0-value from the pitch file. We take this value as a parameter for a call into the light movement function. If it is different from the previous f_0 we found, the light transitions to the new position.

f) Visualisation: For the visualisation, we make the entire screen space available. The y-axis is used as a frequency axis (in the same fashion as a spectrogram) to move the lights. The origin is placed at the centerpoint of the bottom of the screen. Logarithmic scaling of the y-axis accounts for the logarithmic nature of frequencies and their octaves and assures that, e.g., a bassline's notes in the low part of the spectrum are well distinguishable. We cut the upper part of the axis off at around 4 kHz. Higher frequencies occur rather uncommonly in the form of f_0 for typical instruments and their melodies. The x-axis is used for other effects, such as trails of movement, which allows to visually capture certain musical effects, e.g. vibrato. Finally, we create as many light sources as there are CSV files for a particular song. See fig. 1 for a depiction of a running visualisation.

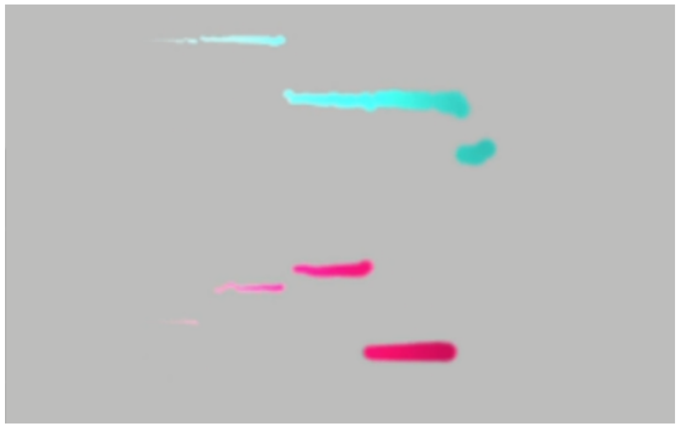

Fig. 1. Example of a running visualisation. Here, two lights visualise the melodies of their respective instruments in the form of a piano roll as a song plays. The trails drift to the left and fade out, and so the lights appear to fly towards the right side of the screen.

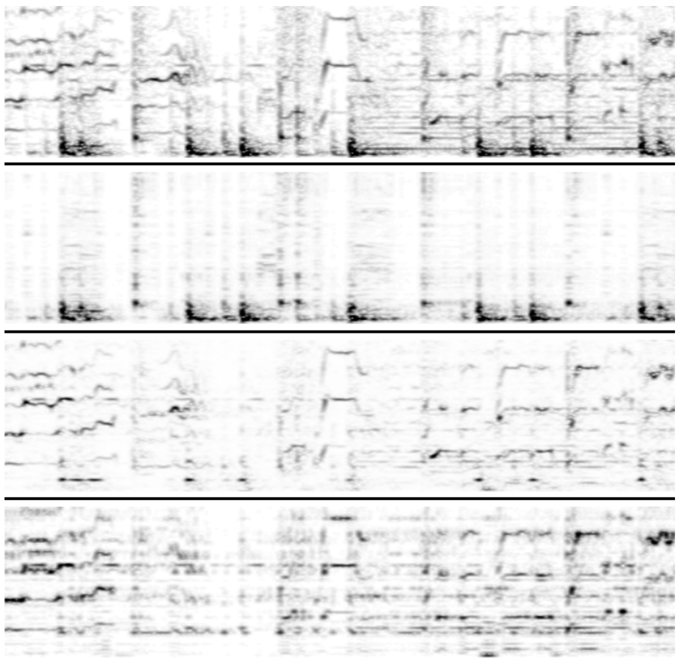

Fig. 2. Spectrogram subtraction example. *Top:* Original spectrogram Y of a modern Western song. *Second row:* Extracted rhythm section. *Third row:* Y after subtraction of said rhythm section, bassline and melodic accompaniment, eventually leaving only the singing voice in (mostly) original format and some left-overs (residues) of the subtracted elements. *Bottom:* By contrast, extracted singing voice from V ($k = 100, i = 500$) with much lower quality.

g) Spectrogram Subtraction: Additionally, for those cases that do prove to be heavily ambiguous for YIN, we use a method called spectrogram subtraction: Instead of working solely on the approximated spectrogram V, we subtract all components corresponding to any unwanted instruments from Y. In our experiments, this procedure usually left one predominant, salient instrument in the original data, with greatly attenuated residues of other instruments. Refer to fig. 2 for an example.

B. System Outline

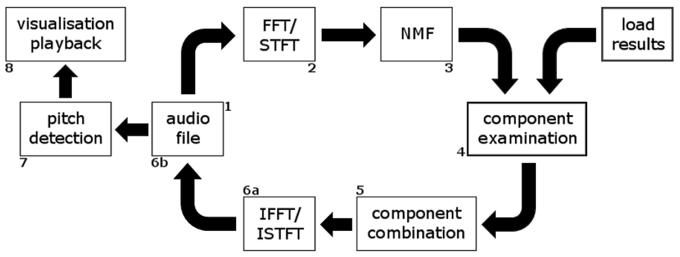

Fig. 3. Outline of our proposed system as an algorithm chain. The numbers indicate the general order of the different steps. It is possible to load previous NMF results and continue from step 4. It is also possible to immediately estimate fundamental frequencies of a given audio file in step 6b.

The individual processing steps of our system are illustrated in fig. 3. First, we extract the spectral features of an audio file using STFT and create its spectrogram. This spectrogram is the input for NMF, which decomposes it into its constituent

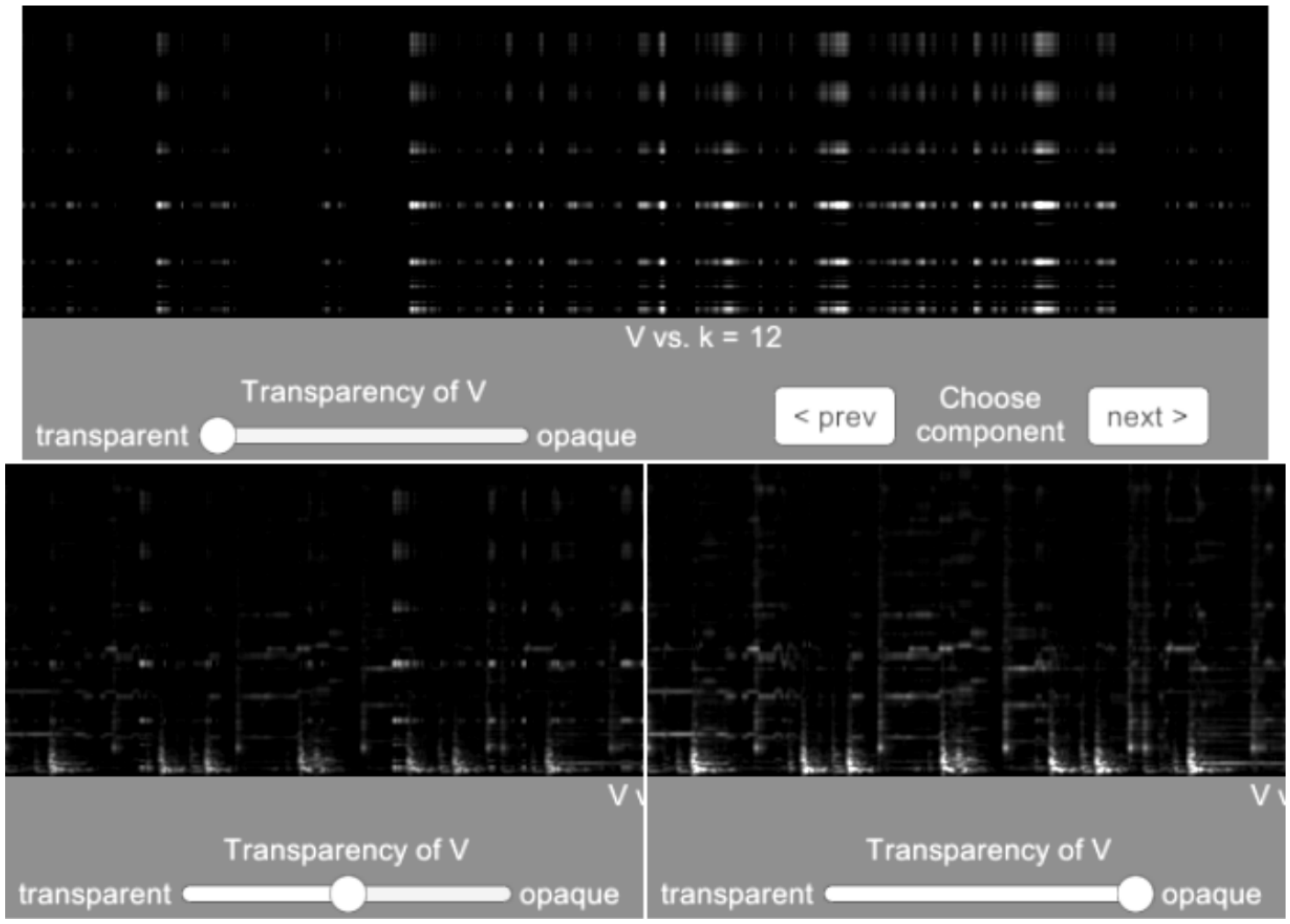

Fig. 4. Component examiner. Here we superimpose energy-normalised components onto V with a controllable degree of transparency. We can easily see the spectral (vertical) and temporal (horizontal) pattern (energy distribution). In this example, $k = 12$ is being evaluated. *Top:* full view of UI and V is invisible (transparent). *Bottom left:* V is 50% transparent. *Bottom right:* Only V is visible, and we can see that this component practically vanishes, indicating that it does not share big proportions of the spectrogram.

parts (or *components*, determined by the dimensionality variable k). These are empirically examined by looking into certain correspondences among them. This examination and verification step is facilitated by a component visualiser we implemented for that matter (see fig. 4), and by listening to the resynthesised audio file. Then, we can combine all components belonging to one (melodic) instrument back together. By doing so, we effectively isolate that instrument from the others. Using resynthesis (ISTFT), we generate an audio signal for the isolated instrument (*sonification*). The YIN algorithm then estimates the fundamental frequencies of this newly generated file for small sequential, non-overlapping time segments, and produces a CSV file (“pitch file”) with frequency entries at the given segments. At this point, we have obtained our desired data and reached the end of the source separation algorithm chain. The process is then repeated from step 4 for the total number of melodic instruments within the original audio file to obtain pitch files for all of them. In the final step, we create as many light sources as there are pitch files. For playback, the pitch files are synchronised with the original audio. As the audio plays back, the lights move up and down, creating an illusion of the music dynamically and directly controlling their movements, in accordance to the melodies played by the instruments they represent.

C. Implementation

We initially implemented our ideas in a Python environment. Since our aim is to create vivid visualisations, we also investigated cross-platform graphics engines. We chose a C#-implementation in Unity in conjunction with third party libraries optimised for task parallelisation and digital signal processing (DSP). Our experiments are conducted with both of these implementations.

In order to verify the suitability of identified music components, we developed a “component examiner”, which enables visual verification of correspondences among the isolated components by superimposing them onto V via an interactively changeable degree of transparency (see fig. 4).

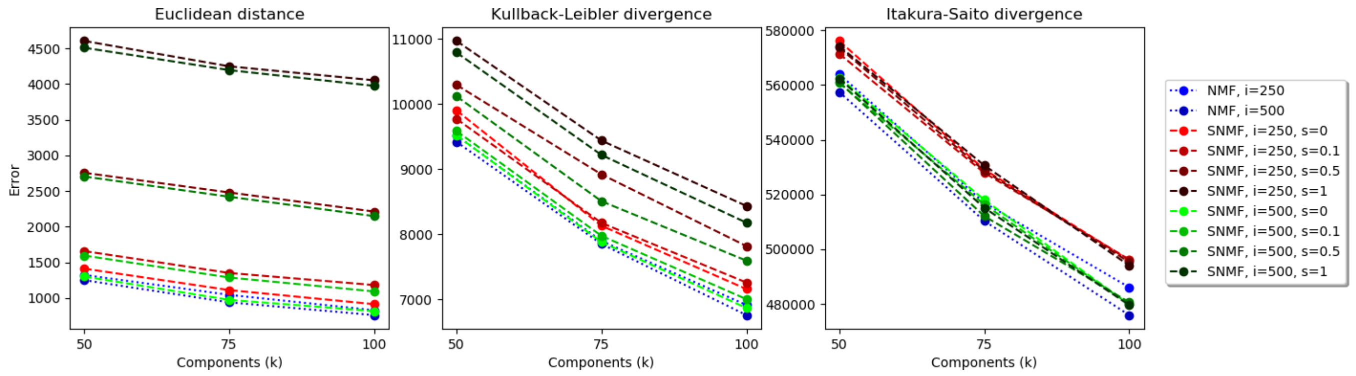

Fig. 5. Comparison of error metrics / cost functions and their effect after i iterations on NMF and SNMF performance with sparsity parameter s, evaluated on a 10-second excerpt of a modern Western song. The values of the different cost functions are not inter-comparable. A sparsity of 0 effectively evaluates to the general NMF algorithm.

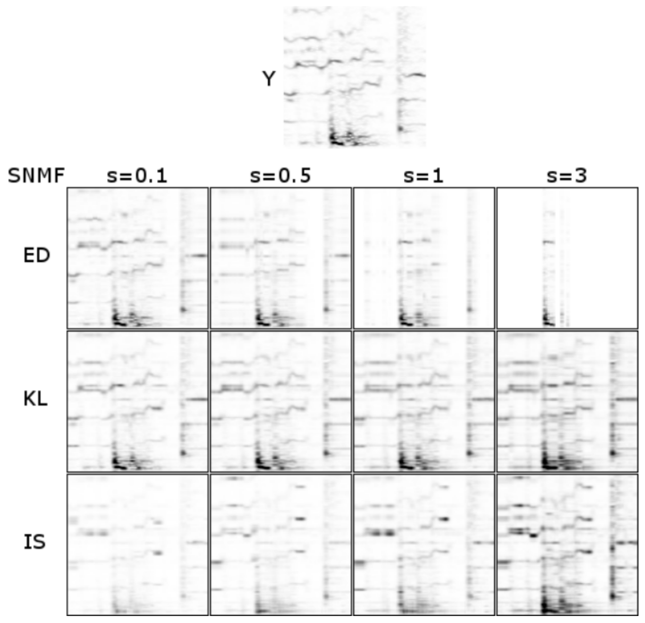

Fig. 6. Effect of the SNMF sparsity parameter s across all error metrics on a spectrogram V created with $k = 100$ and $i = 500$. *Top:* The original spectrogram Y for reference.

D. *Datasets*

For our experiments we created a small dataset consisting of two subsets. One contains short monophonic and polyphonic audio files that we composed ourselves, and from which we know all ground truths such as number of sources, onsets, offsets and exact frequencies. The polyphonic files are also separated into multitracks. The second subset includes small excerpts of less complex modern Western music, from which we can create the ground truth fairly easily. The advantage of this is that for small k, analysing the components can also be performed manually, giving us good insight into the functionality of the algorithms. With larger k this would naturally be impractical. Nonetheless, this knowledge is useful in future work for incorporating (more) automated source counting and separation algorithms.

V. Evaluation and Results

A. *NMF*

While testing different NMF algorithms, we found that the original NMF algorithm converged the fastest in the vast majority of cases and yielded the best results in terms of spectral quality (matrix A) in conjunction with the ED and KL cost functions. See fig. 5 for an overview of comparisons between our test runs with different cost functions while processing a 10-second audio excerpt of a modern Western song. The sparsity parameter s for SNMF unfortunately proved to have an overall negative effect on our goals (see fig. 6).

B. *Findings and Conclusions*

Over the course of development, we were able to ask 21 people for their opinion on the system. All of them agreed in unison that the system works great for the dataset that we completely composed ourselves and for which we have the full ground truth available as multitracks. Naturally, this dataset has the least errors, if any at all. For the dataset with western music and manual annotations, they also agreed in unison that it helps them distinguish between intertwined instruments and even hear things they didn't pay attention to before. This includes, e.g., subliminal accompaniment in the background. However, for the dataset with the same Western music, but sources separated using our system, the experience was less successful, as the miss rate of the pitch estimation did not allow for traceable light movement in accordance to the played melody, leaving many users confused. This observation raises an interesting point: An underperforming system can trigger the exact opposite of our goal. The individual attempt/desire to find correlations between the audio and its visual depiction is apparently so intense that the users at times did not even enjoy the music itself anymore and found the experience stressful.

As we minimise demands to achieve our goals, it is deemed acceptable to allow a few errors inside the system for a very first evaluation of our general idea. At the same time, for the proper evaluation of the user-experience and the reasons stated above, it is important that the miss rate of the pitch estimates is below a certain threshold. Otherwise, the listeners will be confused and lose track of the correlations they have found between light movement and note changes. In our tests we have established that a pitch miss rate of $< 10\%$ is needed to fulfill this criterion, meaning that every tenth note may be a miss, i.e., light movement goes in another direction than expected. For other visualisation ideas, this measure may - and most likely will - be different.

One problem that comes along with our visualisation method is that instruments, which happen to play the same notes, overlap in their lights and become indistinguishable on the 2D plane. This is less of a problem with songs with more rapid note changes. A possible solution could be to reserve designated screen areas for each light, but then it would be a challenge to demonstrate when instruments play the same note simultaneously.

VI. Conclusion and Future Work

We demonstrated that our idea and prototype of a visualisation system was generally well received. Additionally, reproducing earlier research results, we saw that there is a high interest for the enhancement of the listening experience, justifying further endeavours into this. Nonetheless, a few users commented that the experience might quickly become monotonous once the novelty wears off.

Our system naturally leaves much room for improvement particularly on the side of automated source separation. The highest aim would be to create a completely unsupervised machine learning system. One approach could be to adapt clustering methods using non-negative tensor factorisation as presented in [13]. Another approach could involve successively applying algorithms specifically designed for melody extraction [14]. From a given audio signal, one could subtract the main melody first and then the next, less dominant melodic elements, e.g., the accompaniment, perhaps using adapted saliency parameters. This would still leave the problem of real-time capability open. In terms of spectrogram optimisation, there is the possibility of using constant-Q transform [15]. This transform is particularly useful for music signals as it maps the frequency bins efficiently to musical frequencies, producing a transform against log-frequency. In contrast, STFT produces a constant frequency difference per bin. Constant-Q also reduces the number of required frequency bins.

VII. Additional Remarks

A video demonstrating intermediate results can be accessed via https://tinyurl.com/y8wuga56. The video shows first results for a simple self-imposed song with only a few notes and, starting at 3:12 minutes, an example of spectrogram subtraction with 100 components for a famous pop-R&B song.

References

[1] Musicradar, "6 ways to get more out of your DAW's piano roll", October 2014, URL: https://www.musicradar.com/tuition/tech/6-ways-to-get-more-out-of-your-daws-piano-roll-608880.

[2] R. Taylor and P. Boulanger and D. Torres, "Real-time Music Visualization using Responsive Imagery," University of Alberta, Canada, 2007.

[3] M. N. Bain, "Real Time Music Visualization: A Study in the Visual Extension of Music," Ohio State University, USA, 2008.

[4] J. McGowan, G. Leplatre, and I. McGregor, "CymaSense: A Real-Time 3D Cymatics-Based Sound Visualisation Tool," in Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems (DIS '17 Companion). ACM, New York, NY, USA, pp. 270–274, 2017.

[5] K. O'Hanlon, H. Nagano, N. Keriven, and M. D. Plumbley, "Non-negative group sparsity with subspace note modelling for polyphonic transcription," IEEE/ACM Trans. Audio, Speech and Lang. Proc., vol. 24, no. 3, pp. 530–542, 2016.

[6] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," in Nature, vol. 401, pp. 788–791, 1999.

[7] N. J. Bryan, G. J. Mysore and G. Wang, "ISSE: An Interactive Source Separation Editor," in Human Factors in Computing Systems (CHI), Toronto, Canada, April 2014.

[8] P. O. Hoyer, "Non-negative Matrix Factorization with Sparseness Constraints," in J. Mach. Learn. Res., vol. 5, pp. 1457–1469, December 2004.

[9] J. Le Roux, F. J. Weninger and J. R. Hershey, "Sparse NMF - half-baked or well done?," in Mitsubishi Electric Research Laboratories (MERL), TR2015-023, Cambridge, MA, USA, March 2015.

[10] C. Fevotte, N. Bertin and J-L. Durrieu, "Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis," in Neural computation, vol. 21, pp. 793–830, October 2008.

[11] D. W. Griffin and J. S. Lim, "Signal estimation from modified short-time Fourier transform," in IEEE Transaction on Acoustics, Speech, and Signal Processing, vol. ASSP-32, pp. 804–807, May 1983.

[12] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," in J. Acoust. Soc. Am., vol. 111, pp. 1917–1930, 2002.

[13] T. Barker and T. Virtanen, "Non-negative tensor factorisation of modulation spectrograms for monaural sound source separation," in Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, pp. 827–831, 2013.

[14] J. Salamon, E. Gómez, D. Ellis and G. Richard, "Melody Extraction from Polyphonic Music Signals: Approaches, Applications and Challenges," in IEEE Signal Processing Magazine, vol. 31, pp. 118–134, February 2014.

[15] J. C. Brown, "Calculation of a constant Q spectral transform," in J. Acoust Soc. Am., vol. 89, 1991.