Friedrich-Alexander-Universität Erlangen-Nürnberg

Lab Course

Harmonic Percussive Source Separation

International Audio Laboratories Erlangen

Prof. Dr. Meinard Müller
Friedrich-Alexander Universität Erlangen-Nürnberg
International Audio Laboratories Erlangen
Lehrstuhl Semantic Audio Processing
Am Wolfsmantel 33, 91058 Erlangen
meinard.mueller@audiolabs-erlangen.de

International Audio Laboratories Erlangen
A Joint Institution of the
Friedrich-Alexander Universität Erlangen-Nürnberg (FAU) and
the Fraunhofer-Institut für Integrierte Schaltungen IIS

Authors:

Jonathan Driedger
Thomas Prätzlich

Tutors:

Jonathan Driedger
Thomas Prätzlich

Contact:

Jonathan Driedger, Thomas Prätzlich
Friedrich-Alexander Universität Erlangen-Nürnberg
International Audio Laboratories Erlangen
Lehrstuhl Semantic Audio Processing
Am Wolfsmantel 33, 91058 Erlangen
`jonathan.driedger@audiolabs-erlangen.de`
`thomas.praetzlich@audiolabs-erlangen.de`

Lab Course

Harmonic Percussive Source Separation

Abstract

Sounds can broadly be classified into two classes. Harmonic sound on the one hand side is what we perceive as pitched sound and what makes us hear melodies and chords. Percussive sound on the other hand is noise-like and usually stems from instrument onsets like the hit on a drum or from consonants in speech. The goal of harmonic-percussive source separation (HPSS) is to decompose an input audio signal into a signal consisting of all harmonic sounds and a signal consisting of all percussive sounds. In this lab course, we study an HPSS algorithm and implement it in MATLAB. Exploiting knowledge about the spectral structure of harmonic and percussive sounds, this algorithm decomposes the spectrogram of the given input signal into two spectrograms, one for the harmonic, and one for the percussive component. Afterwards, two waveforms are reconstructed from the spectrograms which finally form the desired signals. Additionally, we describe the application of HPSS for enhancing chroma feature extraction and onset detection. The techniques used in this lab cover median filtering, spectral masking and the inversion of the short-time Fourier transform.

1 Harmonic-Percussive Source Separation

When listening to our environment, there exists a wide variety of different sounds. However, on a very coarse level, many sounds can be categorized to belong in either one of two classes: harmonic or percussive sounds. Harmonic sounds are the ones which we perceive to have a certain *pitch* such that we could for example sing along to them. The sound of a violin is a good example of a harmonic sound. *Percussive* sounds often stem from two colliding objects like for example the two shells of castanets. An important characteristic of percussive sounds is that they do not have a pitch but a very clear localization in time. Many real-world sounds are mixtures of harmonic and percussive components. For example, a note played on a piano has a percussive onset (resulting from the hammer hitting the strings) preceding the harmonic tone (resulting from the vibrating string).

Homework Excercise 1

- Think about three real world examples of sounds which are clearly harmonic and three examples of sounds which are clearly percussive.
- What are characteristics of harmonic and percussive signals? Sketch a waveform of a percussive signal and the waveform of a harmonic signal. What are the main differences between those waveforms?

The goal of harmonic-percussive source separation (HPSS) is to decompose a given input signal into a sum of two component signals, one consisting of all harmonic sounds and the other consisting of all percussive sounds. The core observation in many HPSS algorithms is that in a spectrogram representation of the input signal, harmonic sounds tend to form horizontal structures (in time-direction), while percussive sounds form vertical structures (in frequency-direction). For an example, have a look at Figure 1 where you can see the power spectrograms of two signals. Figure 1a shows the power spectrogram of a sine-tone with a frequency of 4000 Hz and a duration of one second. This tone is as harmonic as a sound can be. The power spectrogram shows just one horizontal line. Contrary, the power spectrogram shown in Figure 1b shows just one vertical line. It is the spectrogram of a signal which is zero everywhere, except for the sample at 0.5 seconds

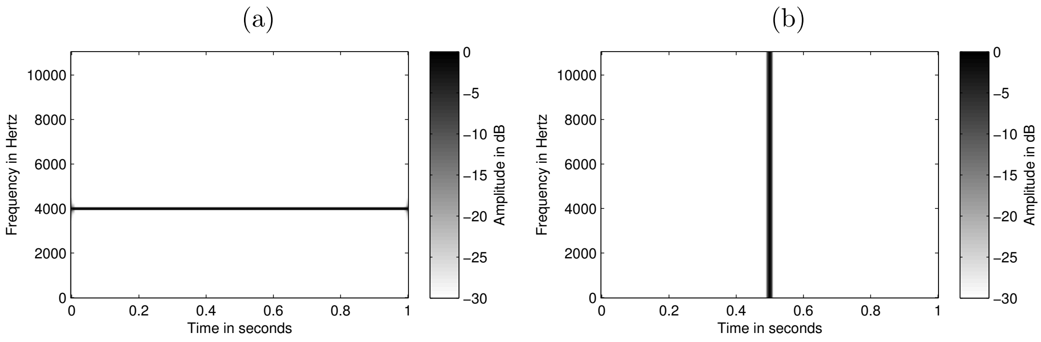

Figure 1: **(a):** Spectrogram of an ideal harmonic signal. **(b):** Spectrogram of an ideal percussive signal.

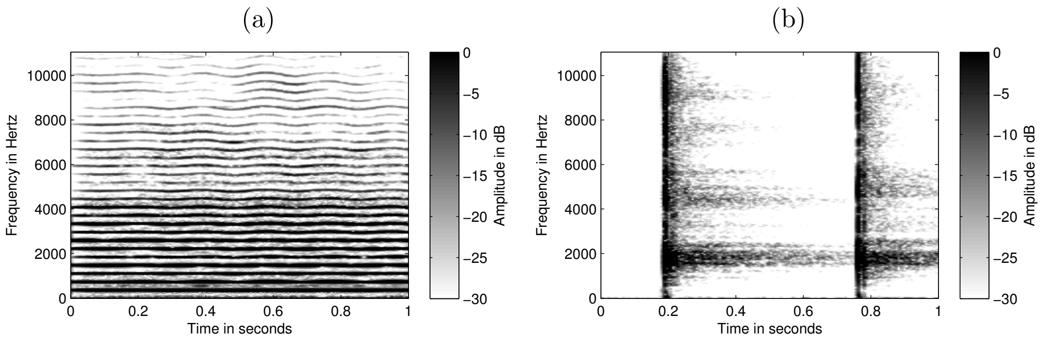

Figure 2: **(a):** Spectrogram of a recording of a violin. **(b):** Spectrogram of a recording of a castanets.

where it is one. Therefore, when listening to this signal, we just hear a brief "click" at 0.5 seconds. This signal is the prototype of a percussive sound. The same kind of structures can be observed in Figure 2, which shows a spectrogram of a violin recording and a spectrogram of a castanets recording.

Real world signals are usually mixtures of harmonic and percussive sounds. Furthermore, there is no absolute definition of when a sound stops "being harmonic" and starts "being percussive". Think, for example, of white noise which cannot be assigned to either one of these classes. However, with the above observations it is possible to decide if a time-frequency instance of a spectral representation of the input signal, like the short-time Fourier transform (STFT), belongs rather to the harmonic component or rather to the percussive component. This can be done in the following way. Assume we want to find out if a time-frequency bin in the STFT of the input signal belongs to the harmonic component. In this case, the bin should be part of some horizontal, and therefore harmonic structure. We can check this by first applying some filter to the power spectrogram of the STFT, which enhances horizontal structures and suppresses vertical structures and see if the filtered bin has some "high value". However, even if its value is high, it might still belong to some even stronger vertical, and therefore percussive structure. We therefore apply another filter to the power spectrogram which enhances vertical structures and suppresses horizontal structures. Now, in the case that the value of our bin in this vertically enhanced spectrogram is lower than in the horizontally enhanced spectrogram, it is very likely that it belongs to some harmonic sound and we can assign it to the harmonic component. Otherwise, if its value was higher in the vertically enhanced spectrogram, we directly know that it is rather part of some percussive sound and assign it to the percussive component. This way, we can decide for every time-frequency instance of the original STFT of the input signal whether it belongs to the harmonic, or to the percussive component and construct two new STFTs. In the STFT for the harmonic component, all bins which were assigned to the percussive component are set to zero, and vice versa for the percussive component. Finally, by "inverting" these STFTs, we get the audio signals for the harmonic and the percussive component.

Homework Excercise 2

- Suppose you apply an HPSS algorithm to white noise. Recall that white noise has a constant power spectral density (it is also said to be flat). What do you expect the harmonic and the percussive component to sound like?
- If you apply an HPSS algorithm to a recording of your favorite rock band. What do you expect the harmonic and the percussive component to sound like?

2 An HPSS Algorithm

We will now describe an actual HPSS algorithm. Formally, given a discrete input audio signal $x : \mathbb{Z} \to \mathbb{R}$, the algorithm should compute a harmonic component signal x_h and a percussive component signal x_p, such that $x = x_h + x_p$. Furthermore, the signals x_h and x_p contain the harmonic and percussive sounds of x, respectively. In the following we describe the consecutive steps of an HPSS algorithm. We start with the computation of the *STFT* (Section 2.1) and proceed with enhancing the power spectrogram using *median filtering* (Section 2.2). Afterwards, the filtered spectrograms are used to compute *binary masks* (Section 2.3) which are used to construct STFTs for the harmonic and the percussive component. These STFTs are finally transformed back to the time domain (Section 2.4).

2.1 Short-Time Fourier Transform

In the first step, we compute the short-time Fourier transform (STFT) $\mathcal{X}$ of the signal x as:

$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N) \tag{1}$$

with $m \in [0 : M-1] := \{0, \ldots, M-1\}$ and $k \in [0 : N-1]$, where M is the number of frames, N is the frame size and length of the discrete Fourier transform, $w : [0 : N-1] \to \mathbb{R}$ is a window function and H is the hopsize. From $\mathcal{X}$ we can then derive the power spectrogram $\mathcal{Y}$ of x:

$$\mathcal{Y}(m,k) := |\mathcal{X}(m,k)|^2. \tag{2}$$

Homework Excercise 3

- The parameters of the STFT have a crucial influence on the HPSS algorithm. Think about what happens to $\mathcal{Y}$ in the case you choose N to be very large or very small. How could this influence the algorithm? (Hint: Think about how N influences the time- and frequency-resolution of the STFT.)
- Explain in technical terms why harmonic sounds form horizontal and percussive sounds form vertical structures in spectrograms (Hint: Have a look at the exponential basis functions of the STFT. What does one of these functions describe? How can an impulse be represented with them).

Lab Experiment 1

- Load an audio file from the `Data` folder using for example `[x,fs]=wavread('CastanetsViolin.wav');`.
- Compute the STFT $\mathcal{X}$ of the input signal x using the provided function `stft.m` with the parameters `N=1024`, `H=512`, `w=win('sin',N)`.
- Compute the power spectrogram $\mathcal{Y}$ according to Equation (2).
- Visualize $\mathcal{Y}$ using the provided function `visualize_matrix.m`. Can you spot harmonic and percussive structures? Note that this function has an optional second argument `lcomp` which can be used to apply a logarithmic compression to the visualized matrix. We recommend using `lcomp=10` when visualizing spectrograms.
- Do the same for the parameters `N=128`, `H=64`, `w=win('sin',N)`, and `N=8192`, `H=4096`, `w=win('sin',N)`. How do the spectrograms change when you change the parameters? What happens to the harmonic and percussive structures?
- Have a look into the provided function code.

2.2 Median Filtering

In the next step, we want to compute a *harmonically enhanced* spectrogram $\tilde{\mathcal{Y}}_h$ and a percussively enhanced spectrogram $\tilde{\mathcal{Y}}_p$ by filtering $\mathcal{Y}$. This can be done by using a *median filter*. The median of a set of numbers can be found by arranging all numbers from lowest to highest value and picking the middle one. E.g. the median of the set $\{7, 3, 4, 6, 5\}$ is 5. Formally, let $A = \{a_n \in \mathbb{R} | n \in [0 : N-1]\}$ be a set of real numbers of size N. Furthermore, we assume without loss of generality that $a_n \leq a_{n'}$ for $n, n' \in [0 : N-1]$, $n < n'$. Then, the median of A is defined as

$$\mathrm{median}(A) := \begin{cases} a_{\frac{N-1}{2}} & \text{for } N \text{ being odd} \\ \frac{1}{2}(a_{\frac{N}{2}} + a_{\frac{N}{2}+1}) & \text{otherwise} \end{cases} \tag{3}$$

Now, given a matrix $B \in \mathbb{R}^{M \times K}$, we define harmonic and percussive median filters

$$\mathrm{medfilt}_h(B)(m,k) := \mathrm{median}(\{B(m-\ell_h,k), \ldots, B(m+\ell_h,k)\}) \tag{4}$$

$$\mathrm{medfilt}_p(B)(m,k) := \mathrm{median}(\{B(m,k-\ell_p), \ldots, B(m,k+\ell_p)\}) \tag{5}$$

for $M, K, \ell_h, \ell_p \in \mathbb{N}$, where $2\ell_h + 1$ and $2\ell_p + 1$ are the lengths of the median filters, respectively. Note that we simply assume $B(m,k) = 0$ for $m \notin [0 : M-1]$ or $k \notin [0 : K-1]$. The enhanced spectrograms are then computed as

$$\tilde{\mathcal{Y}}_h := \mathrm{medfilt}_h(\mathcal{Y}) \tag{6}$$

$$\tilde{\mathcal{Y}}_p := \mathrm{medfilt}_p(\mathcal{Y}) \tag{7}$$

Homework Excercise 4

- The arithmetic mean of a set $A \subset \mathbb{R}$ of size N is defined as $\mathrm{mean}(A) := \frac{1}{N}\sum_{n=0}^{N-1} a_n$. Compute the median and the mean for the set $A = \{2, 3, 190, 2, 3\}$. Why do you think the HPSS algorithm employs median filtering and not mean filtering?
- Apply a horizontal and a vertical median filter of length 3 to the matrix

$$B = \begin{bmatrix} 1 & 1 & 46 & 2 \\ 3 & 1 & 50 & 1 \\ 60 & 68 & 70 & 67 \\ 2 & 1 & 65 & 1 \end{bmatrix}$$

- Explain in your own words why median filtering allows for enhancing/suppressing harmonic/percussive structures in a spectrogram.

Lab Experiment 2

- Apply harmonic and percussive median filters to the power spectrogram $\mathcal{Y}$ which you computed in the previous exercise (`N=1024`, `H=512`, `w=win('sin',N)`) using the provided function `medianFilter.m`.
- Play around with different filter lengths (3, 11, 51, 101).
- Visualize the filtered spectrograms using the function `visualize_matrix.m`. What are your observations?
- Have a look into the provided function code.

2.3 Binary Masking

Having the enhanced spectrograms $\tilde{\mathcal{Y}}_h$ and $\tilde{\mathcal{Y}}_p$, we now need to assign all time-frequency bins of $\mathcal{X}$ to either the harmonic or the percussive component. This can be done by *binary masking.* A binary mask is a matrix $\mathcal{M} \in \{0,1\}^{M \times K}$. It can be applied to an STFT $\mathcal{X}$ by computing $\mathcal{X} \odot \mathcal{M}$, where the operator $\odot$ denotes point-wise multiplication. A mask value of one preserves the value in the STFT and a mask value of zero suppresses it. For our HPSS algorithm, the binary masks are defined by comparing the values in the enhanced spectrograms $\tilde{\mathcal{Y}}_h$ and $\tilde{\mathcal{Y}}_p$.

$$\mathcal{M}_h(m,k) := \begin{cases} 1 & \text{if } \tilde{\mathcal{Y}}_h(m,k) \geq \tilde{\mathcal{Y}}_p(m,k) \\ 0 & \text{else} \end{cases} \tag{8}$$

$$\mathcal{M}_p(m,k) := \begin{cases} 1 & \text{if } \tilde{\mathcal{Y}}_p(m,k) > \tilde{\mathcal{Y}}_h(m,k) \\ 0 & \text{else.} \end{cases} \tag{9}$$

Applying these masks to the original STFT $\mathcal{X}$ yields the STFTs for the harmonic and the percussive component of the signal $\mathcal{X}_h := (\mathcal{X} \odot \mathcal{M}_h)$ and $\mathcal{X}_p := (\mathcal{X} \odot \mathcal{M}_p)$. Note that by the definition of $\mathcal{M}_h$ and $\mathcal{M}_p$, it holds that $\mathcal{M}_h(m,k) + \mathcal{M}_p(m,k) = 1$ for $m \in [0 : M-1]$, $k \in [0 : K-1]$. Therefore, every time-frequency bin of $\mathcal{X}$ is assigned either to $\mathcal{X}_h$ or $\mathcal{X}_p$.

Homework Excercise 5

Assume you have the two enhanced spectrograms

$$\tilde{\mathcal{Y}}_h = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 3 & 1 & 1 \\ 60 & 68 & 68 & 67 \\ 1 & 2 & 1 & 1 \end{bmatrix}, \quad \tilde{\mathcal{Y}}_p = \begin{bmatrix} 1 & 1 & 46 & 1 \\ 3 & 1 & 50 & 2 \\ 2 & 1 & 65 & 1 \\ 2 & 1 & 65 & 1 \end{bmatrix}$$

Compute the binary masks $\mathcal{M}_h$ and $\mathcal{M}_p$ and apply them to the matrix

$$\mathcal{X} = \begin{bmatrix} 1 & 1 & 46 & 2 \\ 3 & 1 & 50 & 1 \\ 60 & 68 & 70 & 67 \\ 2 & 1 & 65 & 1 \end{bmatrix}$$

Lab Experiment 3

- Use the median filtered power spectrograms $\tilde{\mathcal{Y}}_h$ and $\tilde{\mathcal{Y}}_p$ from the previous exercise (filter length 11) to compute the binary masks $\mathcal{M}_h$ and $\mathcal{M}_p$.
- Visualize the masks using the function `visualize_matrix.m` (this time without logarithmic compression).
- Apply the masks to the original STFT $\mathcal{X}$ to compute $\mathcal{X}_h$ and $\mathcal{X}_p$.
- Visualize the power spectrograms $\mathcal{Y}_h$ and $\mathcal{Y}_p$ of $\mathcal{X}_h$ and $\mathcal{X}_p$ using `visualize_matrix.m`.

2.4 Inversion of the Short-Time Fourier Transform

In the final step, we need to transform our constructed STFTs $\mathcal{X}_h$ and $\mathcal{X}_p$ back to the time-domain. To this end, we apply an "inverse" STFT to these matrices to compute the component signals x_h and x_p. Note that the topic "inversion of the STFT" is not as trivial as it might seem at the first glance. In the case that $\mathcal{X}$ is the original STFT of an audio signal x, and further preconditions are satisfied (for example that $N \geq H$ for N being the size of the discrete Fourier transform and H being the hopsize of the STFT), it is possible to invert the STFT and to reconstruct x from $\mathcal{X}$ perfectly. However, as soon as the original STFT $\mathcal{X}$ has been modified to some $\tilde{\mathcal{X}}$, for example by masking, there might be no audio signal which has exactly $\tilde{\mathcal{X}}$ as its STFT. In such a case, one usually aims to find an audio signal whose STFT is "approximately" $\tilde{\mathcal{X}}$. See Section 4 for pointers to the literature. For this Lab Course, you can simply assume that you can invert the STFT using the provided MATLAB function `istft.m`.

Homework Excercise 6

Assume $\mathcal{X}$ is the original STFT of some audio signal x. Why do we need the precondition $N \geq H$ for N being the size of the discrete Fourier transform and H being the hopsize of the STFT to reconstruct x from $\mathcal{X}$ perfectly?

Lab Experiment 4

- Apply the inverse STFT function `istft.m` to X_h and X_p from the previous experiment and listen to the results.
- Save the computed harmonic and percussive component by using `wavwrite(x_h,fs,'harmonicComponent.wav');` and `wavwrite(x_p,fs,'percussiveComponent.wav');`

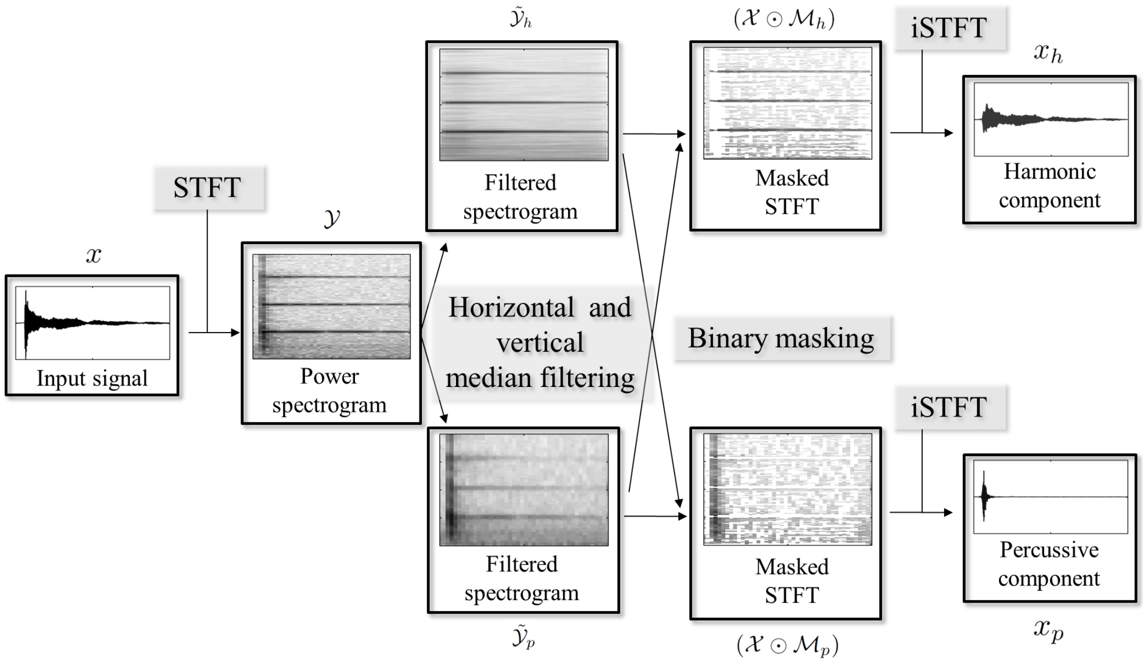

Figure 3: Harmonic-percussive source separation.

2.5 Physical Interpretation of Parameters

Note that one can specify the filter lengths of the harmonic and percussive median filters in seconds and Hertz, respectively. This makes their physical interpretation easier. Given the sampling rate f_s of the input signal x as well as the frame length N and the hopsize H, we can convert filter lengths given in seconds and Hertz to filter lengths given in indices

$$L_h(t) := \left\lceil \frac{f_s}{H} t \right\rceil \tag{10}$$

$$L_p(d) := \left\lceil \frac{N}{f_s} d \right\rceil \tag{11}$$

Homework Excercise 7

Assume $f_s = 22050$ Hz, $N = 1024$, and $H = 256$. Compute $L_h(0.5 \text{ sec})$ and $L_p(600 \text{ Hz})$.

Lab Experiment 5

- Complete the implementation of the HPSS algorithm in `HPSS.m`:
  1. Compute the STFT $\mathcal{X}$ of the input signal x using the provided function `stft.m`.
  2. Compute the power spectrogram $\mathcal{Y}$ from the $\mathcal{X}$.
  3. Convert the median filter lengths from seconds and Hertz to indices using the Equations (10) and (11).
  4. Apply median filters to $\mathcal{Y}$ using the provided function (`medianFilter.m`) to compute $\mathcal{Y}_h$ and $\mathcal{Y}_p$.
  5. Derive the masks $\mathcal{M}_h$ and $\mathcal{M}_p$ from $\mathcal{Y}_h$ and $\mathcal{Y}_p$.
  6. Compute $\mathcal{X}_h$ and $\mathcal{X}_p$.
  7. Apply the inverse STFT (`istft.m`) to get x_h and x_p.
- Test your implementation:
  1. Load the audio files `'Stepdad.wav'`, `Applause.wav`, and `DrumSolo.wav` from the `Data` folder.
  2. Apply `[x_h,x_p]=HPSS(x,N,H,w,fs,lh_sec,lp_Hz)` using the parameters `N=1024`, `H=512`, `w=win('sin',N)`, `lh_sec=0.2`, and `lp_Hz=500` to all loaded signals.
  3. Listen to the results.

3 Applications of HPSS

In many audio processing tasks, the essential information lies in either the harmonic or the percussive component of an audio signal. In such cases, HPSS is very well suited as a pre-processing step to enhance the outcome of an algorithm. In the following, we introduce two procedures that can be improved by applying HPSS. The harmonic component from the HPSS algorithm can be used to enhance chroma features (Section 3.1) and the percussive component helps to improve the results of an onset detection procedure (Section 3.2).

3.1 Enhancing Chroma Features using HPSS

Two pitches sound similar when they are an octave apart from each other (12 tones in the equal tempered scale). We say that these pitches share the same chroma which we refer to by the pitch spelling names $\{\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \mathrm{D}^\sharp, \mathrm{E}, \mathrm{F}, \mathrm{F}^\sharp, \mathrm{G}, \mathrm{G}^\sharp, \mathrm{A}, \mathrm{A}^\sharp, \mathrm{B}\}$. Chroma features exploit the above observation, by adding up all frequency bands in a power spectrogram that belong to the same chroma. Technically this can be realized by the following procedure. First we assign a pitch index (MIDI pitch number) to each frequency index $k \in [1 : N/2 - 1]$ of the spectrogram by using the formula:

$$p(k) = \mathrm{round}\left(12 \log_2\left(\frac{k \cdot f_s}{440 \cdot N}\right)\right) + 69. \tag{12}$$

where N is the number of frequency bins in the spectrogram and f_s is the sampling rate of the audio signal. Note that p maps frequency indices corresponding to frequencies around the chamber tone A4 (440 Hz) to its MIDI pitch number 69. Then we add up all frequency bands in the power spectrogram belonging to the same chroma $c \in [0 : 11]$:

$$\mathcal{C}(m, c) := \sum_{\{k \mid p(k) \bmod 12 = c\}} \mathcal{Y}(m, k) \tag{13}$$

where $m \in [0 : M - 1]$ and M is the number of frames.

Chroma features are correlated with the pitches and the harmonic structure of music. Pitches usually form horizontal structures in the spectrogram, whereas transient or percussive sounds form vertical structures. Percussive sounds have a negative impact on the chroma extraction, as they "activate all frequencies" in the spectrogram, see also Homework 3. Hence, one way to improve the chroma extraction is to first apply HPSS and to perform the chroma extraction on the power spectrogram of the harmonic component signal $\mathcal{Y}_h(m,k) = |\mathcal{X}_h(m,k)|^2$, see also Exercise 6.

Lab Experiment 6

Apply the HPSS algorithm as a pre-processing step in a chroma extraction procedure:

1. Load the file `CastanetsViolin.wav` using `[x,fs]=wavread('CastanetsViolin.wav')`.
2. Compute chroma features on x using the provided implementation in `simple_chroma.m` with the parameters `N=4410` and `H=2205`.
3. Visualize the chroma features by using the visualization function given in `visualize_simpleChroma.m`.
4. Apply your HPSS algorithm to separate the castanets from the violin.
5. Use the harmonically enhanced signal x_h to compute chroma features and visualize them.
6. Now compare the visualization of the chroma extracted from the original signal x and the chroma extracted from the harmonic component signal x_h. What do you observe?

3.2 HPSS for Onset Detection

Onset detection is the task of finding the temporal positions of note onsets in a music recording. More concrete, the task could be to detect all time positions on which some drum is hit in a recording of a rock song. One way to approach this problem is to assume, that drum hits emit a short burst of high energy and the goal is therefore to detect these bursts in the input signal. To this end, one first computes the *short-time power* $\mathcal{P}$ of the input signal x by

$$\mathcal{P}(m) := \sum_{n=0}^{N-1} x(n + mH)^2 \tag{14}$$

where H is the hopsize and N is the length of one frame (similar to the computation of the STFT). Since we are looking for time-positions of high energy, the goal is therefore to detect peaks in $\mathcal{P}$. A common technique to enhance peaks in a sequence is to subtract the *local average* $\tilde{\mathcal{P}}$ from $\mathcal{P}$ itself. $\tilde{\mathcal{P}}$ is defined by

$$\tilde{\mathcal{P}}(m) := \sum_{j=-J}^{J} \mathcal{P}(m+j)\frac{1}{2J+1} \tag{15}$$

for a neighborhood $J \in \mathbb{N}$, $m \in [0 : M-1]$, and M is the number of frames. Note that we assume $\mathcal{P}(m) = 0$ for $m \notin [0 : M-1]$. From this, we compute a *novelty curve* $\mathcal{N}$

$$\mathcal{N}(m) := \max(0, \mathcal{P}(m) - \tilde{\mathcal{P}}(m)) \tag{16}$$

The peaks in $\mathcal{N}$ indicate positions of high energy in x, and are therefore potential time positions of drum hits.

This procedure works well in case the initial assumption, namely that onsets or drum hits emit some burst of energy which stand out from the remaining energy in the signal, is met. However, especially in professionally mixed music recordings, the short-time energy is often adjusted to be more or less constant over time (compression). One possibility to circumvent this problem is to apply HPSS to the input signal prior to the onset detection. The onset detection is then executed solely on the percussive component which usually contains all drum hits and satisfies the assumption of having energy bursts at the respective time-positions.

Lab Experiment 7

- Complete the implementation of the onset detection algorithm in `onsetDetection.m`:
  1. Compute the short-time power $\mathcal{P}$ of the input signal x using the provided function `stp.m`.
  2. Compute the local average $\tilde{\mathcal{P}}$ as defined in Equation (15). (Hint: Note that Equation (15) can be formulated as a convolution and that you can compute convolutions in MATLAB using the command `conv`. Note further that this command has an option `'same'`. Finally, have a look at the MATLAB command `ones`).
  3. Compute the novelty curve $\mathcal{N}$ as described in Equation (16).
- Test your implementation by applying it to the audio file `StillPluto_BitterPill.wav`. As a starting point, use $N = 882$, $H = 441$, and $J = 10$.
- Sonify your results using the function `sonify_noveltyCurve.m`. This function will generate a stereo audio signal in which you can hear the provided original signal in one of the channels. In the other channel, each peak in the provided novelty curve is audible as a click sound. You can therefore check by listening whether the peaks in your computed novelty curve are aligned with drum hits in the original signal. To apply the function `sonify_noveltyCurve.m`, you need to specify the sampling frequency of the novelty curve. How can you compute it? (Hint: It is dependent on H and the sampling frequency f_s of the input audio signal).
- Listen to the generated results. What is your impression?
- Now apply your HPSS algorithm to the audio file and rerun the detection algorithm on just the percussive component x_p. Again, sonify the results. What is your impression now?

4 Further Notes

The task of decomposing an audio signal into its harmonic and its percussive component has received large research interest in recent years. This is mainly because for many applications it is useful to consider just the harmonic or the percussive portion of an input signal. Harmonic-percussive separation has been applied to many audio processing tasks, such as audio remixing [10], the enhancement of chroma features [12], tempo estimation [5], or time-scale modification [1, 3]. Several decomposition algorithms have been proposed. In [2], the percussive component is modeled by detecting portions in the input signal which have a rather noisy phase behavior. The harmonic component is then computed by the difference of the original signal and the computed percussive component. The algorithms presented in [11] and [4] both exploit the spectral structure of harmonic and percussive sounds that we have seen in this lab course. The HPSS algorithm discussed in this lab is the one presented in [4].

Concerning the task of inverting a modified STFT, one can say that it is not possible in general from a mathematical point of view. This is the case since the space of signals is smaller than the space of STFTs and therefore no bijective mapping between the two spaces can exist. However, it is possible to "approximate" inversions, see [6].

If you are interested in further playing around with chroma features or onset detection (and their applications) you can find free MATLAB implementations at [9] and [7].

Finally we would like to also point out that median filtering techniques have also successfully applied to other signal domains. They can for example be used to reduce certain classes of noise, namely *salt and pepper noise*, in images, see [8].

References

[1] Jonathan Driedger, Meinard Müller, and Sebastian Ewert. Improving time-scale modification of music signals using harmonic-percussive separation. *Signal Processing Letters, IEEE*, 21(1):105–109, 2014.

[2] Chris Duxbury, Mike Davies, and Mark Sandler. Separation of transient information in audio using multiresolution analysis techniques. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, Limerick, Ireland, 12 2001.

[3] Chris Duxbury, Mike Davies, and Mark Sandler. Improved time-scaling of musical audio using phase locking at transients. In *Audio Engineering Society Convention 112*, 4 2002.

[4] Derry Fitzgerald. Harmonic/percussive separation using medianfiltering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, 2010.

[5] Aggelos Gkiokas, Vassilios Katsouros, George Carayannis, and Themos Stafylakis. Music tempo estimation and beat tracking by applying source separation and metrical relations. In *ICASSP*, pages 421–424, 2012.

[6] Daniel W. Griffin and Jae S. Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32(2):236–243, 1984.

[7] Peter Grosche and Meinard Müller. Tempogram Toolbox: MATLAB tempo and pulse analysis of music recordings. In *12th International Conference on Music Information Retrieval (ISMIR, late-breaking contribution)*, Miami, USA, 2011.

[8] S. Jayaraman, T. Veerakumar, and S. Esakkirajan. *Digital Image Processing.* Tata McGraw Hill, 2009.

[9] Meinard Müller and Sebastian Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 215–220, Miami, FL, USA, 2011.

[10] Nobutaka Ono, Kenichi Miyamoto, Hirokazu Kameoka, and Shigeki Sagayama. A real-time equalizer of harmonic and percussive components in music signals. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 139–144, Philadelphia, Pennsylvania, USA, 2008.

[11] Nobutaka Ono, Kenichi Miyamoto, Jonathan LeRoux, Hirokazu Kameoka, and Shigeki Sagayama. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. In *European Signal Processing Conference*, pages 240–244, Lausanne, Switzerland, 2008.

[12] Yushi Ueda, Yuuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based approach for automatic chord detection using refined acoustic features. In *ICASSP*, pages 5518–5521, 2010.