SGN-14007 Introduction to Audio Processing **Veikko Nopanen 263400**

Project topic 1: Separation of drums from music signals

Source code available: https://github.com/venopan/sgn-14007

In this project I will try to separate harmonic and percussive components apart from each other from a monaural audio signal. Algorithms used in this project are based on a paper by Nobutaka Ono et al. [1] which are implemented in Python. Results are shown as signal-to-noise ratio evaluation and signal power spectrograms. Additionally, separated components (10 iterations) are saved as audio files and can be listened to compare with the original audio.

My assumption in this project is that no signal can be separated to harmonic and percussive components perfectly. There will be artifacts and some parts of the different components will be mixed. The results will depend on the number of iterations as well.

The algorithm itself consist of a few simple steps. Signal is processed in short windows, so short-term Fourier transform is used. The harmonic and percussive components are initially set as half of the range-compressed spectrum of the original signal. Components are updated using a formula and depending on the number of maximum iterations allowed the updated components get more accurate. Finally, the results are binarized and converted to waveforms again using inversed STFT.

Evaluation is done by calculating the signal-to-noise ratio of the original signal and the separated one. So, both harmonic and percussive component gets evaluated. Evaluation results are calculated using audio files 'police03short.wav' and 'project_test1.wav' which had been provided with the project assignment. Results are shown below.

'police03short.wav':

10 iterations:

```
Harmonic SNR: 4.0, Percussive SNR: 3.2
```

50 iterations:

```
Harmonic SNR: 3.8, Percussive SNR: 3.1
```

'project_test1.wav':

10 iterations

```
Harmonic SNR: 7.0, Percussive SNR: 1.4
```

50 iterations

```
Harmonic SNR: 7.6, Percussive SNR: 1.1
```

The amount of iterations used seems to have some effect over the SNR.

Figure 1 and 2 show the spectrograms of the audio files used. First there's the original range-compressed power spectrogram and the others are the harmonic and percussive component ones. I used 50 iterations for all spectrograms.

'police03short.wav':

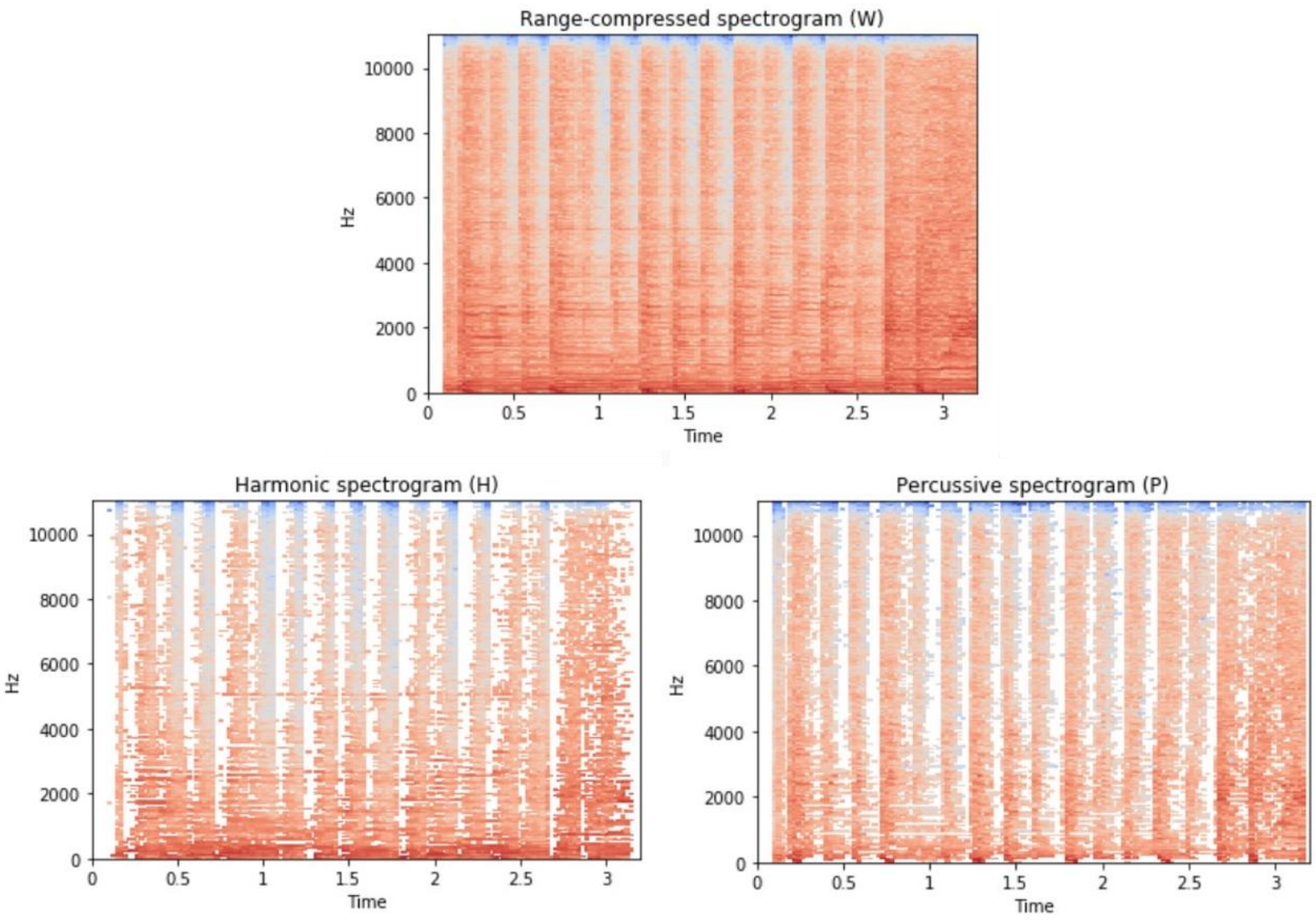

Figure 1: Spectrograms of 'police03short.wav'

‘project_test1.wav’:

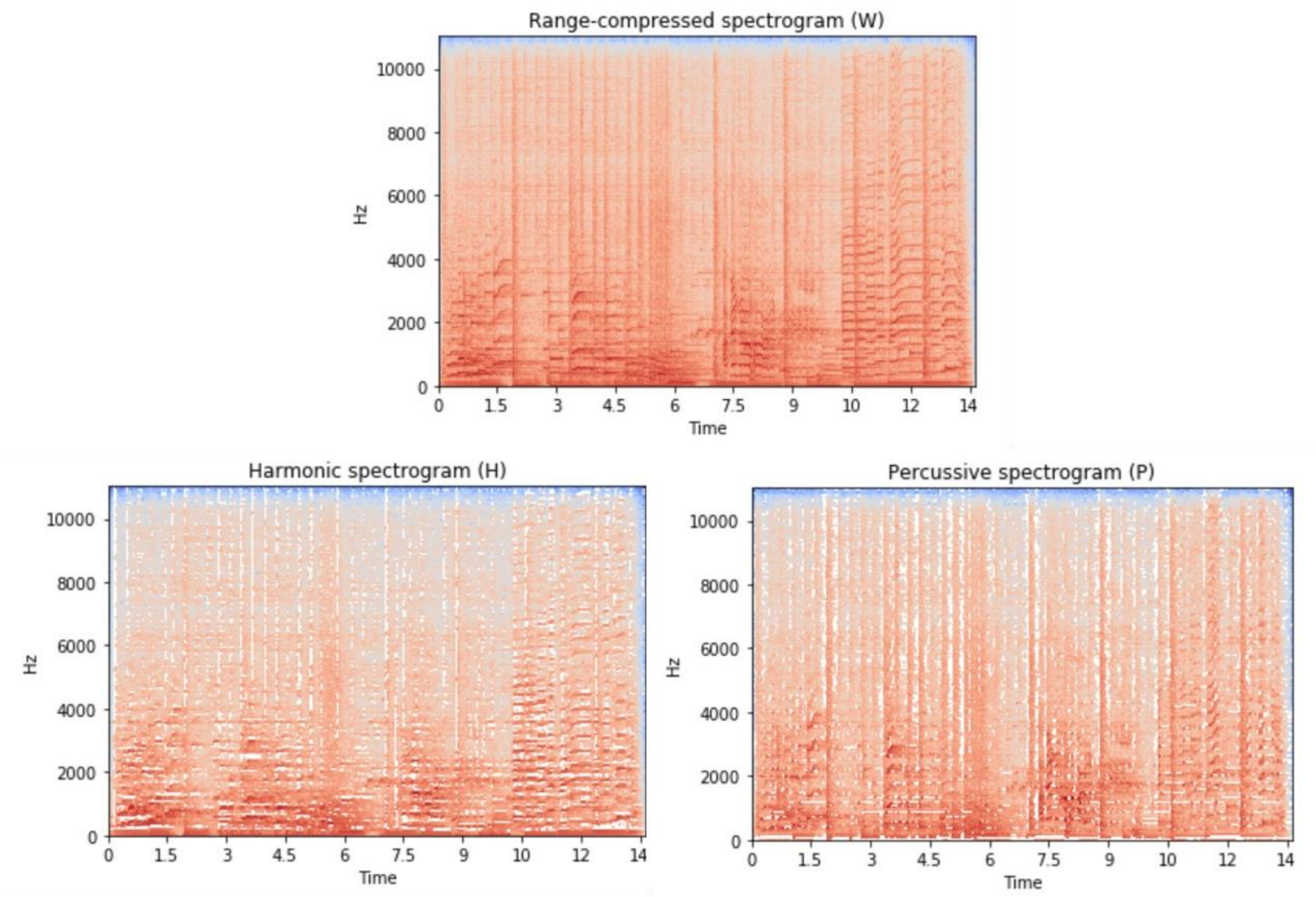

Figure 2: Spectrograms of ‘project_test1.wav’

Figure 3 shows one part (from 6 s to 10 s) of the ‘project_test1.wav’ file where singing can be seen on the lower end of the spectrograms as smooth waves, around 60-1500 Hz [2]. Singing is also present in percussive component and it can be also heard from the generated file. Otherwise the algorithm does its job well.

‘project_test1.wav’ from 6 s to 10 s:

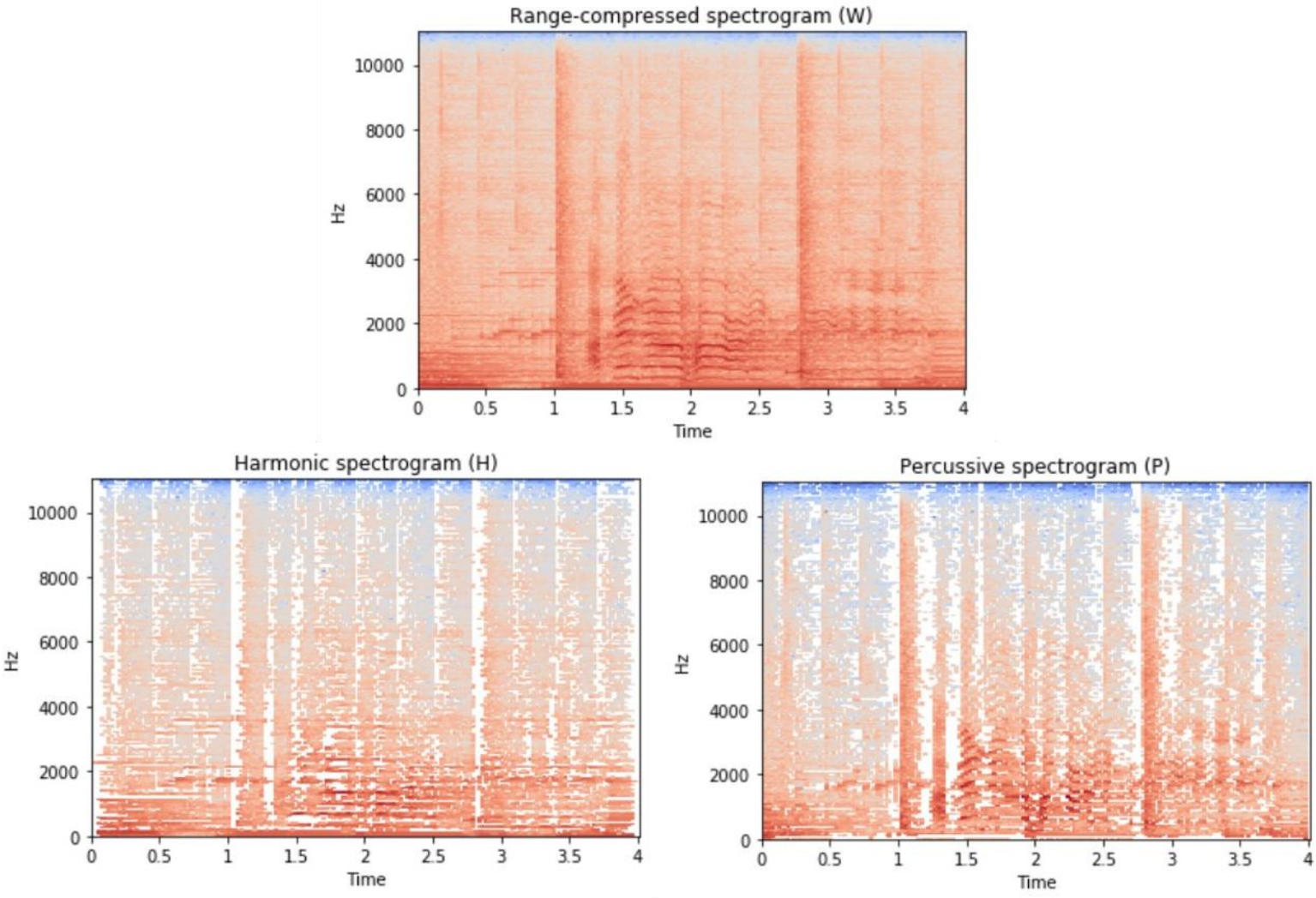

Figure 3: Spectrograms of the singing part

Because of the way that the algorithm works some of the harmonic components that have high attack phase were incorrectly categorized as percussive components. On the other hand, some of the percussive elements that have a smooth envelope like bass drum are easily thought as harmonic components in this implementation.

Based on the results I assume that the algorithm works best with songs that have no singing and the harmonic and percussive are clearly heard separately. The tempo of the song could affect the results too so that slower paced songs perform better.

In conclusion I’m very satisfied on the efficiency of the algorithm. It’s very cool to see these things work when you’ve implemented something on your own. Only thing bothering is the computing performance of the algorithm which seems very slow. I assume that it could be implemented in another, more efficient way.

Sources:

1. N. Ono, K. Miyamoto, J. L. Roux, H. Kameoka and S. Sagayama, “Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram,” in Proc. EUSIPCO, 2008
2. https://newt.phys.unsw.edu.au/jw/voice.html (Accessed 18.12.2019)