Spectrum technical requirements & specification

François Brachais (*deqyra)*

January 22, 2019

Contents

1 Purpose

Spectrum is a software project aiming at applying image processing to sound. In order to do so, Spectrum takes in any audio file, and transforms the waveform into a spectrogram. Since spectral data can be regarded as an image, it is possible to apply image processing techniques to that spectrogram. It is then turned back into an audio waveform.

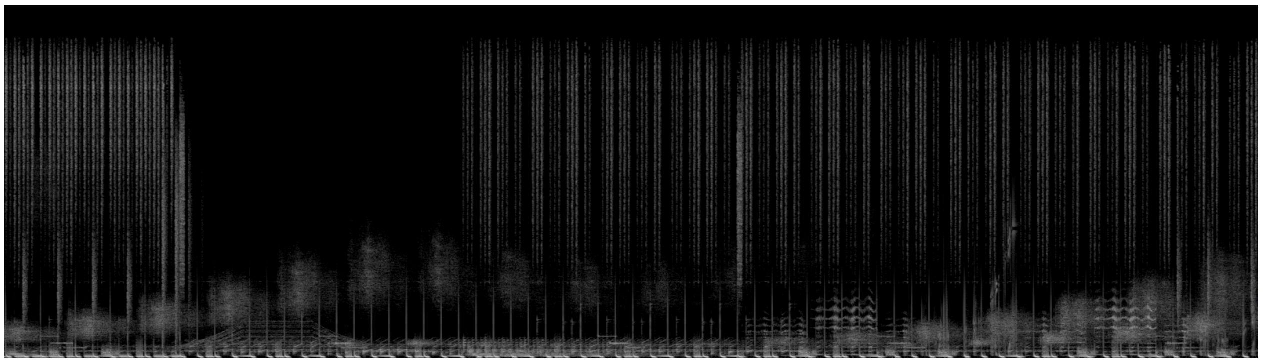
Spectrum aims at enabling users to create otherwise unachievable sound effects that can be reused for any purpose: music production, sound design, etc. It could also be a playful way to discover more about sound transformation and to train one's ear to recognize frequency ranges, and so on.

2 Fourier transform and spectrograms

According to Fourier, any given signal which is periodic over time can be decomposed in an infinite sum of sines of different frequencies: one fundamental frequency, and several multiples of that frequency, which are called harmonics. The weight of each sine in the sum is described by two pieces of information: its amplitude and its phase. In a regular Cartesian coordinate system, the amplitude can be described as the y factor of that sine, while the phase can be described as an x offset. Since we are dealing with sines, the phase actually is an angle ranging in $[-\pi; \pi]$.

As such, an infinite sum of sines characterized by these two parameters (amplitude and phase) can be regarded as an infinite sum of sines with complex coefficients, the amplitude of a sine being represented by the module of the coefficient, and the phase being its argument. Such an operation, i.e. passing over from the time domain (original signal) to the frequency domain (sum of sines) is called the Fourier Transform (FT). It is a reversible operation, which means any complex sum of sines can be turned back into a signal. The reversibility of the Fourier transform is at the stake of the process by which Spectrum is able to use imaging techniques on sound.

Considering an audio file, it is possible to apply FT to small, regular fragments of the waveform. Doing so along the whole length of the audio yields an overview of all of the frequencies present over time in the audio. Such a result is called a spectral image, spectral data, or, more conveniently, a spectrogram. Here follows a spectrogram:



The x axis, just as in the original waveform, is time; the y axis represents frequencies. A white point in the spectrogram means there is a lot of amplitude to that particular frequency at that particular time in the audio, while a black point means that frequency is basically absent from the audio at that time. As can be seen on the image, points can take any shade between black and white.

3 Spectrographical manipulation

As shown above, it is possible to represent a spectrogram with a grey-scale image, which makes it suitable for graphical transformation. Once the image has been transformed, it is still an image, and can still be regarded as a spectrogram which can be turned back into sound. Such transformations, performed in the frequency, may sound totally unique once reverted to the time domain, and may simply be impossible (or incredibly complicated) to achieve directly in the time domain.

For example, the above spectrogram is rich with hi-hat cymbals. If we were to isolate the sound of such cymbals, the corresponding spectrogram would be a straight, white vertical line (many of which can be seen above) on a black background. We decide to apply a "skewing" transformation to that image, so that the straight white line is not vertical anymore, but oblique, from lower left to upper right for example. We then decide to turn that back into audio. How does that sound? It is a frequency sweep from the mid-highs to the highs.

Achieving that result solely in the time domain is impossible. The closest, straight-forward way for this would require some form of spectral delay, which basically boils down to spectrogram generation. This project aims at hearing the effects of different image manipulations: skewing, blurring, twisting, compressing, etc.

4 System specifications

The following basic requirements were established, as part of the core API of the project (a.k.a a headless implementation):

- The software must be able to read audio files in at least Wave format (*.wav files).
- The software must be able to generate the spectrogram of input signals.
- A number of transformations may be available to use, primarily targeting images (spectrograms) but ultimately audio too.
- The end user may be able to tweak the parameters of the transformations.
- The software must be able to reconstruct audio data from the generated spectrogram (and ultimately any kind of image).
- Finally, the software must be able to export the reconstructed sound as an audio file in one of the handled formats.

5 References

Documents that helped me towards getting a very basic idea of what I wanted, and enabled me to start-up the project:

- `https://web.archive.org/web/20141213140451/https://ccrma.stanford.edu/courses/422/projects/WaveFormat/`: A backup of a comprehensive and detailed page about the binary layout of the Wave file format.
- `http://support.ircam.fr/docs/AudioSculpt/3.0/co/Signal%20Representation.html`: A very useful resource containing everything there is to know about audio perception and representation.
- `http://support.ircam.fr/docs/AudioSculpt/3.0/co/Signal%20Analysis.html`: A most valuable resource containing everything there is to know about signal analysis and alternative representations.
- `https://stackoverflow.com/questions/39295589/creating-spectrogram-from-wav-using-fft-in-java`: A very insightful and beautifully explained answer about common troubleshooting when implementing FT algorithms.
- `https://dsp.stackexchange.com/questions/3406/reconstruction-of-audio-signal-from-its-absolute-spec`
`3410#3410`: An answer explaining how to use Griffin & Lim's algorithm to reconstruct audio from a spectrogram.