Asynchronous Activity 3: Computing the Spectrogram of Music

Prof. Jordan C. Hanson

January 21, 2022

Abstract

This activity contains code that will help you display the spectrogram of music. A spectrogram is a graph of time (x-axis), frequency (y-axis), and power (colorscale). This activity is more open-ended than the previous two asynchronous activities, but you will learn a lot by studying the associated code.

1 The Song: For an Angel by Paul van Dyk

Navigate to the course Moodle page and under Unit 2, download the `.wav` file with the song title. Store the file in the same location that you will run the code that analyzes it.

2 The Code: Spectrogram Trance

Navigate to the course Moodle page and under Unit 2 code folder, download the file `spectrogram_trance.m`. What follows is a description of how the code works.

The code begins by reading in the audio track as a `.wav` file, prepares to save every 2nd sample, defines the end of the recording, and takes only the left track of the stereo data:

```
%Read in the audio audio track
[audio_data sampling_rate] = audioread('ForAnAngel.wav');
%Take every n-th sample of audio data:
decimation_factor = 2;
%Specify percentage of song to analyze
stop_point = 0.2;
%Take only one of the two tracks, every n-th sample
audio_data = audio_data(1:decimation_factor:end,1);
```

The first line calls `audioread()` to collect the data from disk, along with the sampling rate. The decimation factor is used at the end of the code block to skip every other sample. In the last line, the `audio_data` variable is also called like (..., 1) to take only the first column and not the second column. The stop point is used below.

```
sampling_rate = sampling_rate/decimation_factor;
time_difference = 1/sampling_rate;
critical_frequency = sampling_rate/2.0;
n_samples = length(audio_data);
stop_sample = floor(stop_point*n_samples);
%Analyze only up to a given sample, based on stop_point
audio_data = audio_data(1:stop_sample);
times = transpose([0:stop_sample]*time_difference);
```

The sampling rate is changed by the decimation factor since we are not using every sample. The time difference between samples is the inverse of the new sampling rate, and the critical frequency is 50% of the new sampling rate. The last sample we will analyze is given by the product of the stop point fraction (0.2 above) and the number of samples. The audio waveform may be plotted against the variable `times`, but it's difficult to see the frequency structure.

3 Playing the Clip

Paul van Dyk is a world-famous trance producer. For an Angel was a track originally released in 1994, and has been remixed and released countless times. The following code will play the song up to the stop point defined above:

```
%Play the clip
player1 = audioplayer(audio_data,sampling_rate,16);
play(player1)
```

4 Spectrogram of the Music

A *spectrogram* is a tool used to visualize digital data, binned into time bins and frequency bins. The following code defines the bins, and computes the *short-time Fourier transform* (STFT) of the audio track:

```
%Calculate the spectrogram
n_window = 2048;
n_inc = 1024;
[spectral_data, info] = stft(audio_data,n_window,n_inc,n_window);
spectral_data = spectral_data(1:end/2,:);
```

The variable `n_window` is the number of samples to use in the current FFT. Next, the code jumps forward `n_inc`, and takes the FFT of that data. The process repeates, producing many FFTs, one for each time step. The last line of the code block removes the redundant negative frequencies, which are represented by half of the data. In the code below, the size of the spectrogram data is obtained, and used to create a vector of frequencies and times to match the dimensions. The `linspace` function is used to make the times and frequencies at regular steps. For the frequencies, there are `n_freq` frequencies between 0 Hz and the critical frequency, for example.

```
[n_freq n_time] = size(spectral_data);
frequencies = linspace(0,critical_frequency,n_freq);
%Get the stop time
stop_time = stop_sample*time_difference;
times_2 = linspace(0,stop_time,n_time);
%Get spectral data into dB (power)
spectral_data_dB = 10*log10(abs(spectral_data));
spectral_data_dB = spectral_data_dB-min(min(spectral_data_dB));
x = [min(times_2) max(times_2)];
y = [min(frequencies) max(frequencies)];
```

After the times and frequencies are created, the spectrogram data is converted to decibels. The variables x and y are used in graphing the data below.

5 Graphing the Data

The following code produces three figures when the code is run in Octave.

```
figure(1)
image(x,y,spectral_data_dB)
xlabel('Time (seconds)')
ylabel('Frequency (Hz)')
h = colorbar()
colormap('ocean')
set(gca(),'fontsize',18,'fontname','Calibri')
set(h,'fontsize',18,'fontname','Calibri')
axis([0 10 0 10000])

figure(2)
image(x,y,spectral_data_dB)
xlabel('Time (seconds)')
ylabel('Frequency (Hz)')
h = colorbar()
colormap('ocean')
set(gca(),'fontsize',18,'fontname','Calibri')
set(h,'fontsize',18,'fontname','Calibri')
axis([0 50 0 400])

figure(3)
image(x,y,spectral_data_dB)
xlabel('Time (seconds)')
ylabel('Frequency (Hz)')
h = colorbar()
colormap('ocean')
set(gca(),'fontsize',18,'fontname','Calibri')
set(h,'fontsize',18,'fontname','Calibri')
axis([50 60 0 8000])
```

The first figure contains the first 10 seconds of the song. There is a frequency sweep followed by a band-limited rhythm. Can you identify in the graph when the band-limited bass kick begins?

The second figure is the same as the first, but for the first 50 seconds. At a certain point, the band-limiting effect is released, and we hear the full power of the bass kick. Can you identify in the figure when this happens?

The third figure contains the spectrogram between 50 and 60 seconds. There is a alarm-like tone that has many Fourier modes (repeating stripes in the figure). However, it rises in pitch gradually. How does this affect the Fourier modes?

6 Your Task: Run the Code and Produce the Figures

Your only job with this asynchronous activity is to run the code, hear the music, and see the effects visualized in the spectrogram! However, once you understand what is going on, you should modify the code to explore the rest of the song. The middle and ending of the song has synthesized piano melodies, and the kick is not always there. See if you can creat interesting graphs highlighting the different sounds in the music. Good luck!