Lab 1

Michael Eller

February 1, 2016

Introduction

The goal of this lab is to explore different methods of categorizing music into specific genres. Several methods of automated music analysis exist already, including **score following**, **automatic music transcription**, **music recommendation**, and **machine listening**. Unfortunately, all of these tools are still rudimentary. Faster, more efficient, methods still need to be developed in order to allow automated music analysis to be implemented on portable music players. The goal of this lab is to develop tools to automatically extract some defining features of music that will help us to categorize them more easily.

1 Sampling Rates

Current high-quality audio standards have a 24-bit depth and is sampled at 96 kHz. This 24-bit depth means there are 16,777,216 possible values for the audio signal at any given instance. It also means we are able to replicate frequencies up to 48 kHz. While this is far above the 20 kHz limit of human hearing, DVD audio is simply not high enough quality enough for a dolphin to listen to.

Assignment

1. Dolphins can only hear sounds over the frequency range [7 - 120] kHz. At what sampling frequency f_s should we sample digital audio signals for dolphins?

Nyquist theorem states that in order to accurately recreate a signal with maximum frequency $f_s/2$, we must sample at a minimum frequency of f_s. Therefore, our sampling frequency for digital audio signals for dolphins should be at minimum 240 kHz. Currently, one of the highest portable audio standards of BD-ROM LPCM (lossless) allow for 24 bit/sample and a maximum sampling frequency of only 192 kHz. Less common standards do exist though: Digital eXtreme Definition at 352.8 kHz using for recording and editing Super Audio CDs, SACD at 2,822.4 kHz known as Direct Stream Digital, and Double-Rate DSD at 5,644.8 kHz used in some professional DSD recorders. Therefore, as it stands, we might want to stay away from producing audio for dolphins (at least until better recording standards are developed.

2 Audio signal

Assignment

2. Write a MATLAB function that extract T seconds of music from a given track. You will use the MATLAB function ~~waveread~~ audioread to read a track and the function play to listen to the track.

   In the lab you will use T =**24 seconds** from the middle of each track to compare the different algorithms. Download the files, and test your function.

Listing 1: extractSound.m

```
function [ soundExtract,p ] = extractSound( filename, time )
%extractSound Extracts time (in seconds) from the middle of the song
%   Write a MATLAB function that extract T seconds of music from a
%   given track. You will use the MATLAB function audioread to
%   read a track and the function play to listen to the track.
info = audioinfo(filename);
[song,~]=audioread(filename);
if time >= info.Duration
    soundExtract=song;
    p=audioplayer(soundExtract,info.SampleRate);
    return;
elseif time<= 1/info.SampleRate
    error('Too small of a time to sample');
end
samples=time*info.SampleRate;
soundExtract=song(floor(info.TotalSamples/2)-floor(samples/2):1: ...
    floor(info.TotalSamples/2)+floor(samples/2));
p=audioplayer(soundExtract,info.SampleRate);
end
```

This MATLAB function is fairly straightforward. MATLAB's built-in function *audioinfo* provided all the necessary attributes to allow my function to parse any .wav file and extract the number of samples needed.

3 Low Level Features

The bulk of music analysis is done in the frequency spectrum, however, there are some low level features that can be found from the time-domain analysis of music as well.

3.1 Loudness

There is not an easy way to describe *loudness*, but one way to mathematically quantize it is by finding it standard deviation, $\sigma(n)$.

$$\sigma(n) = \sqrt{\frac{1}{N-1}\sum_{m=-N/2}^{N/2-1}[x(n+m)-\mathbb{E}[x_n]]^2} \quad \text{with} \quad \mathbb{E}[x_n] = \frac{1}{N}\sum_{m=-N/2}^{N/2-1} x(n+m) \tag{1}$$

Assignment

3. Implement the loudness and ZCR and evaluate these features on the different music tracks. Your MATLAB function should display each feature as a time series in a separate figure.

The following is an excerpt from my *loudness* function. My *extractSound* function extracts 24 seconds around the middle of the song. The song excerpt is then split into frames of size 255 overlapped (at least) halfway with the previous frame.

Listing 2: loudness.m

```
30 [y,~]=extractSound( filename, time );
31 frames_data = buffer(y,frameSize,ceil(frameSize/2));
32 loudness_data=std(frames_data,0,1);
```

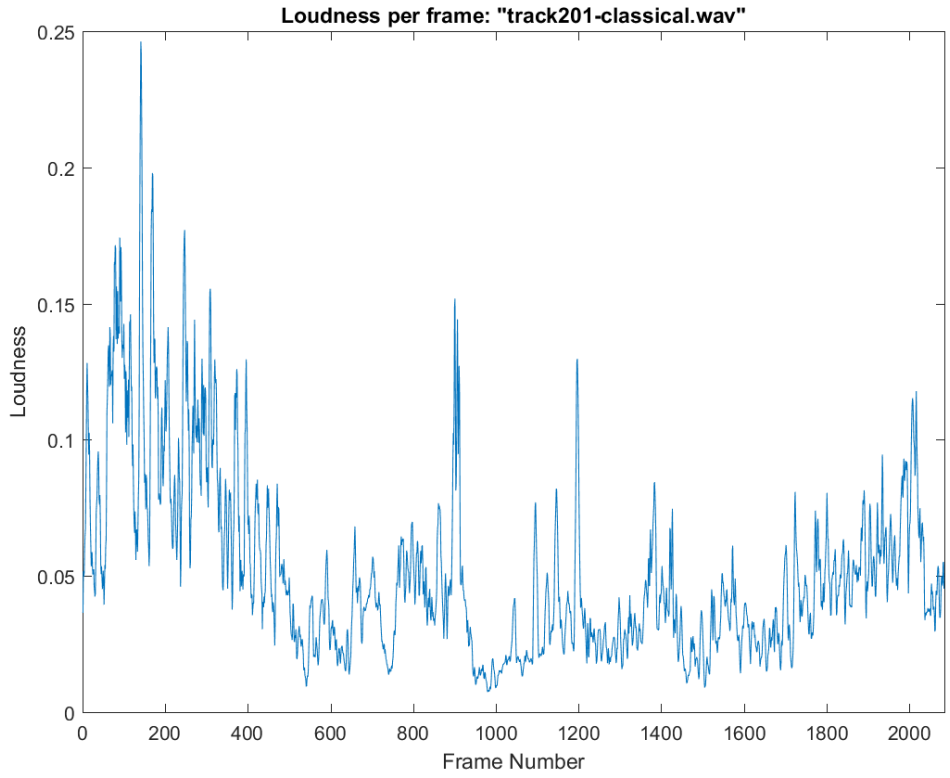

Figure 1: Loudness value per frame

Figure 1 shows an example of the function applied to "track201-classical.wav". The figure would imply that, for the selection of the song, the beginning is quite loud, then quiets down until two loud moments, and finishes at a moderately quiet section. If one listens to the song

selection in question, the results of my loudness function can be confirmed. The other loudness plots can be found in Appendix A.1

A Figures

A.1 Loudness

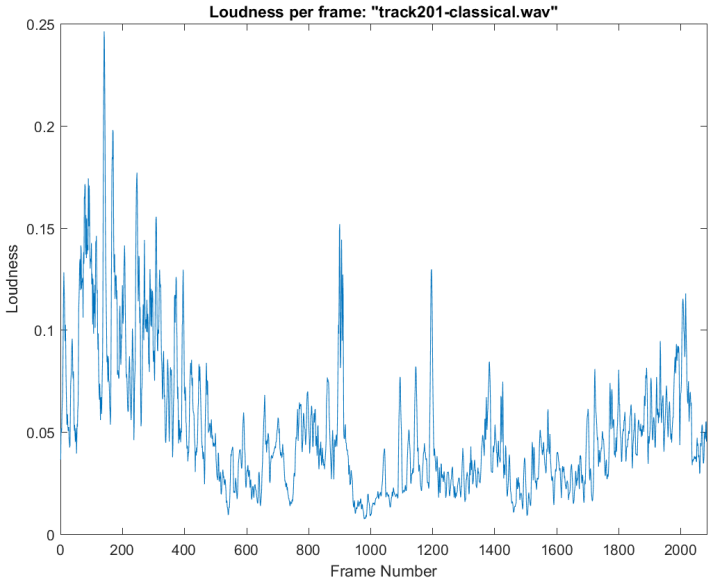

Figure 1: Loudness value per frame, classical201

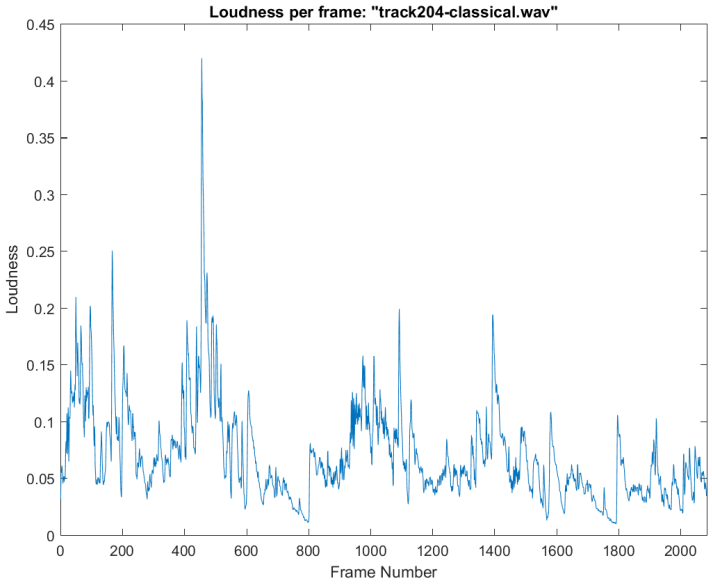

Figure 2: Loudness value per frame, classical204

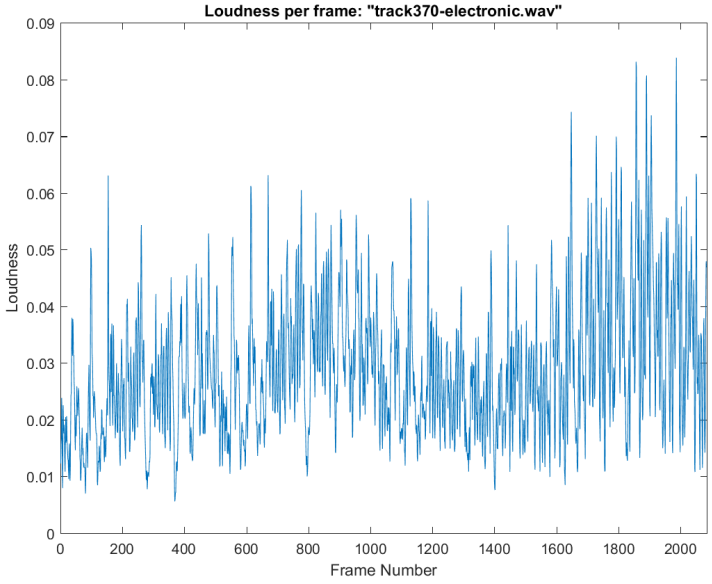

Figure 3: Loudness value per frame, electronic370

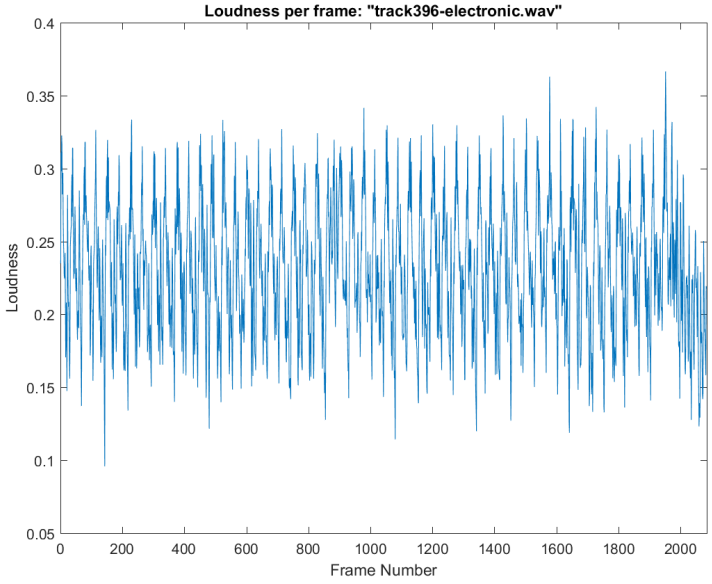

Figure 4: Loudness value per frame, electronic396

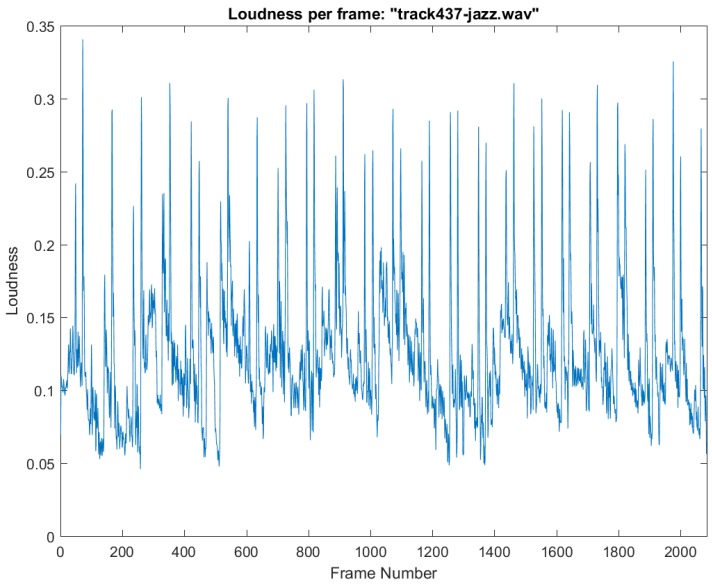

Figure 5: Loudness value per frame, jazz437

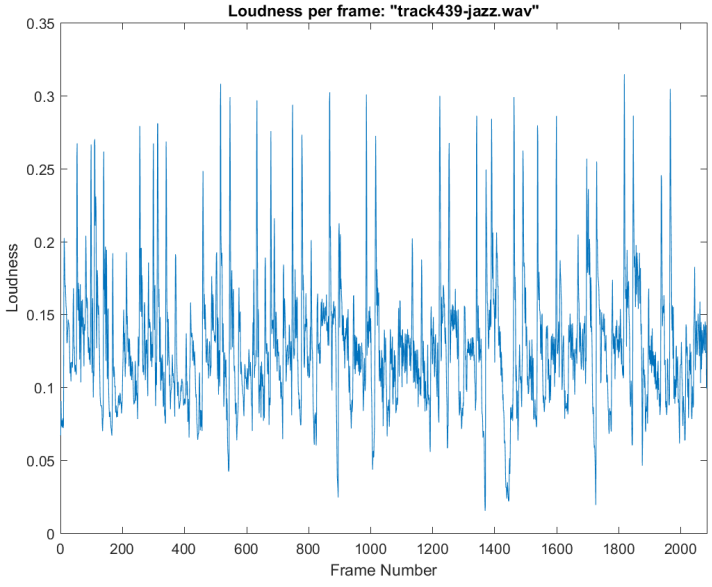

Figure 6: Loudness value per frame, jazz439

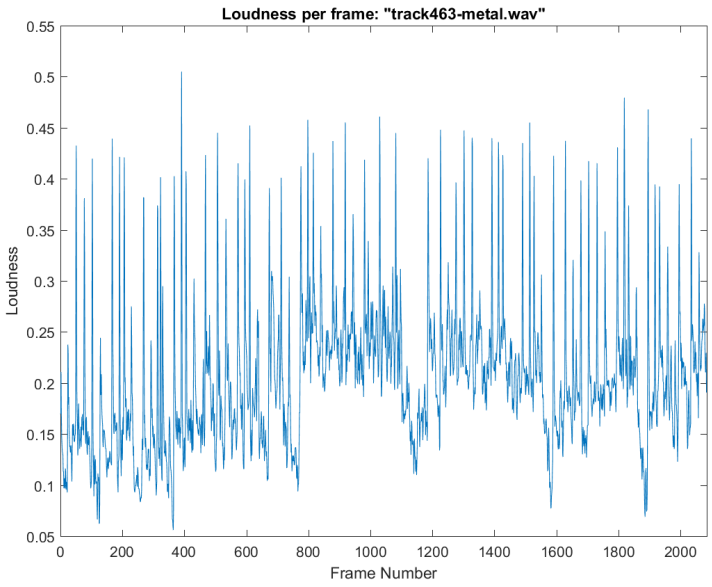

Figure 7: Loudness value per frame, metal463

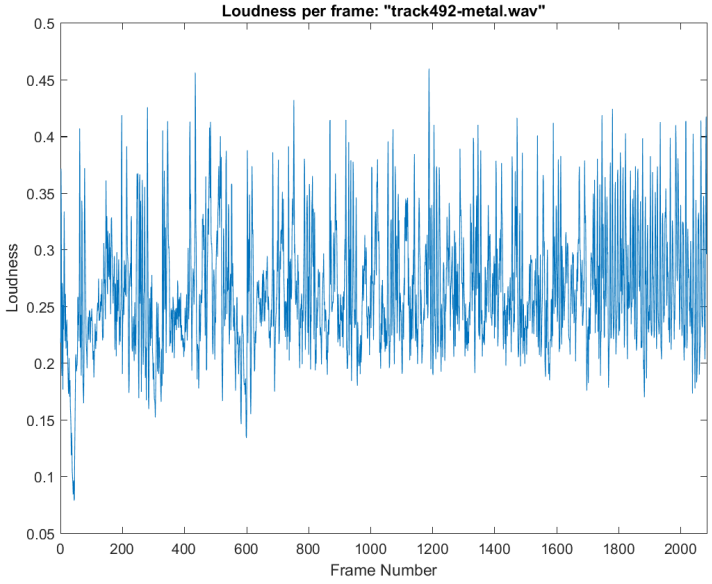

Figure 8: Loudness value per frame, metal492

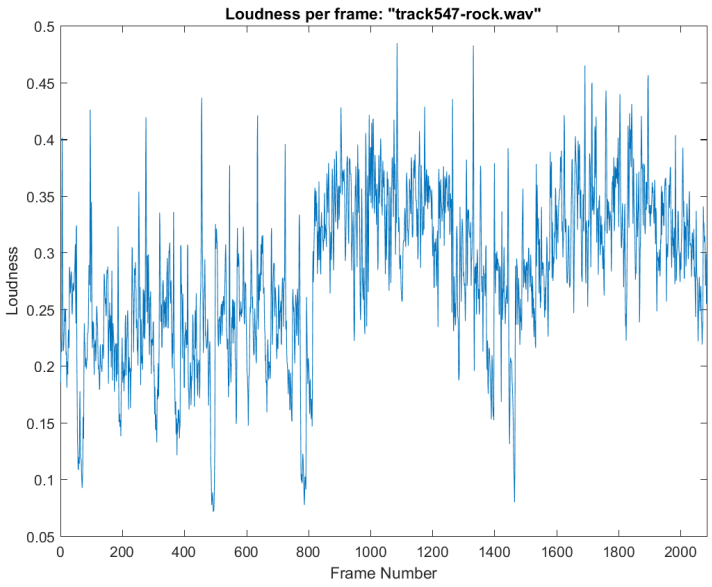

Figure 9: Loudness value per frame, rock547

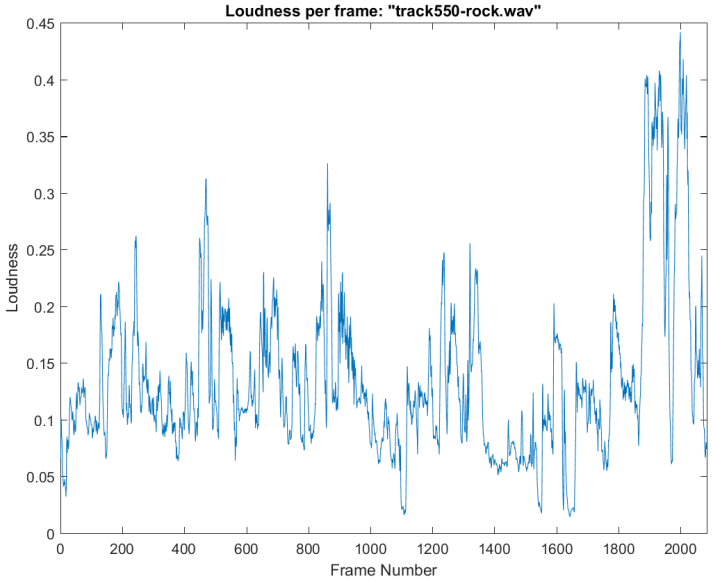

Figure 10: Loudness value per frame, rock550

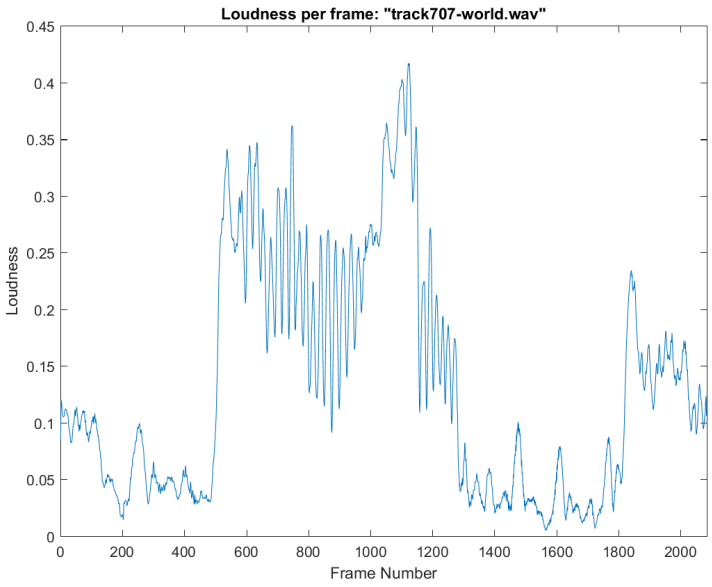

Figure 11: Loudness value per frame, world707

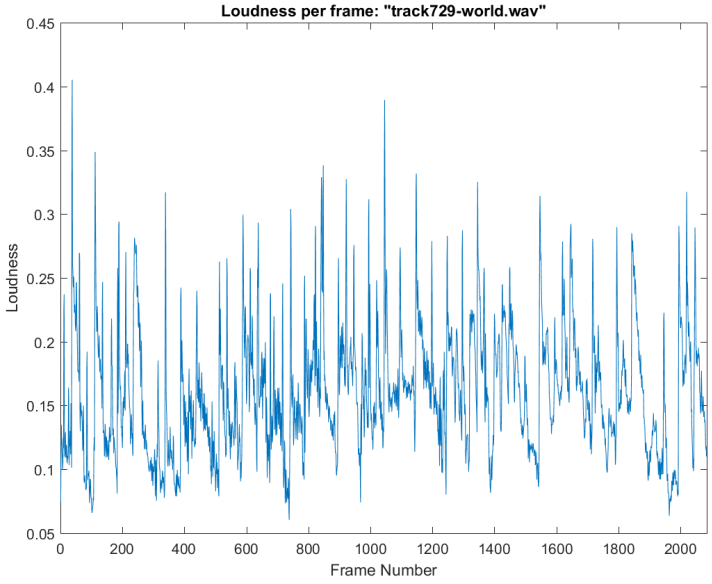

Figure 12: Loudness value per frame, world729