project_update_1113

November 13, 2024

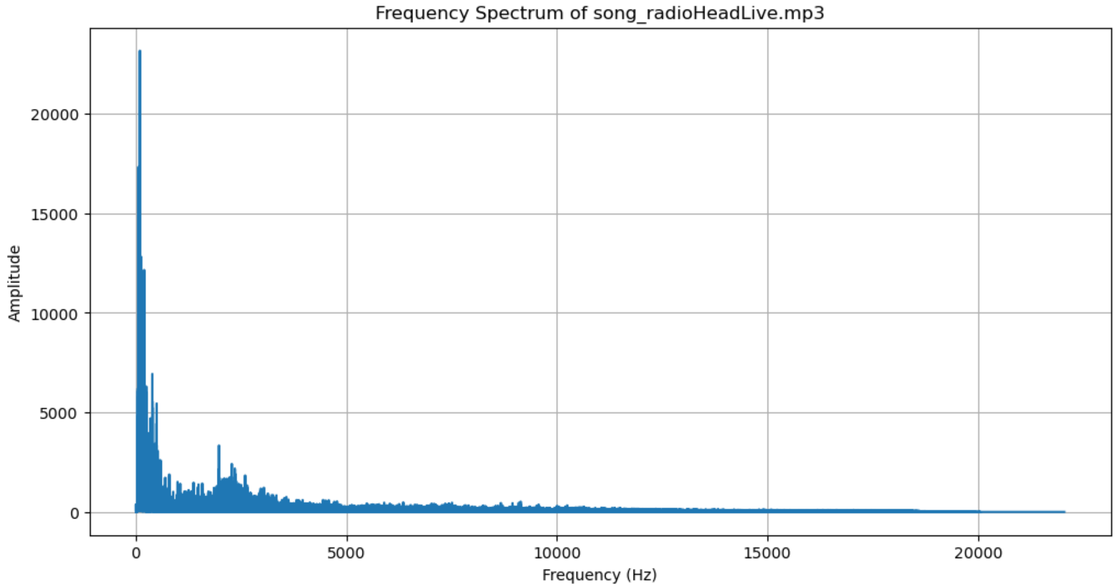
0.1 For Wednesday, November 13, please list what you proposed to have completed by the middle of Week 7, and summarize what you have accomplished. You can be as detailed as you like. This counts towards your preliminary report score.

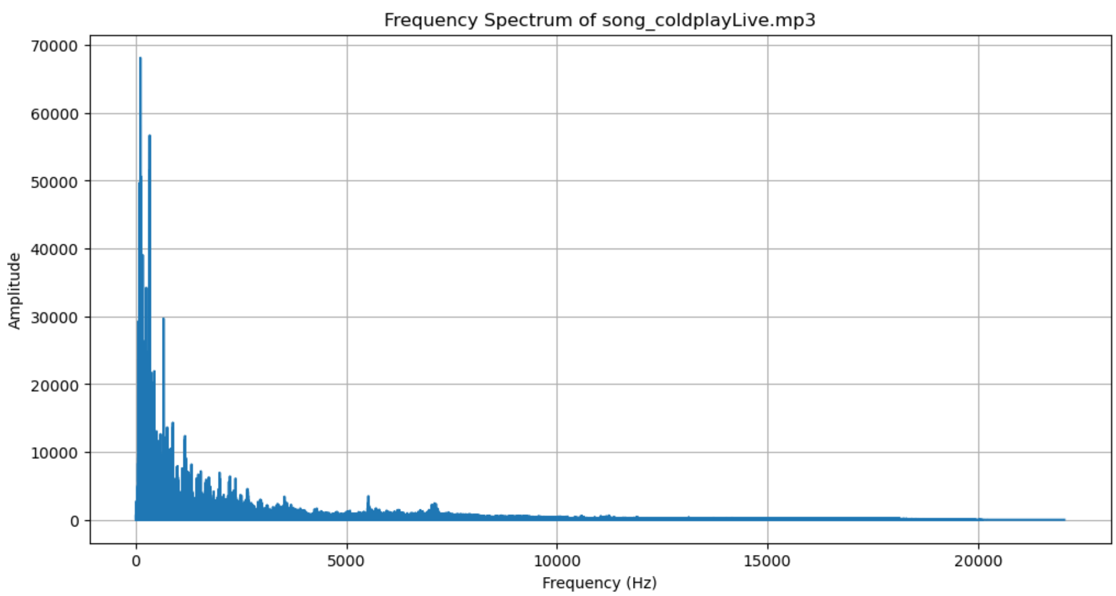
0.2 Proposed to be done by the end of the week:

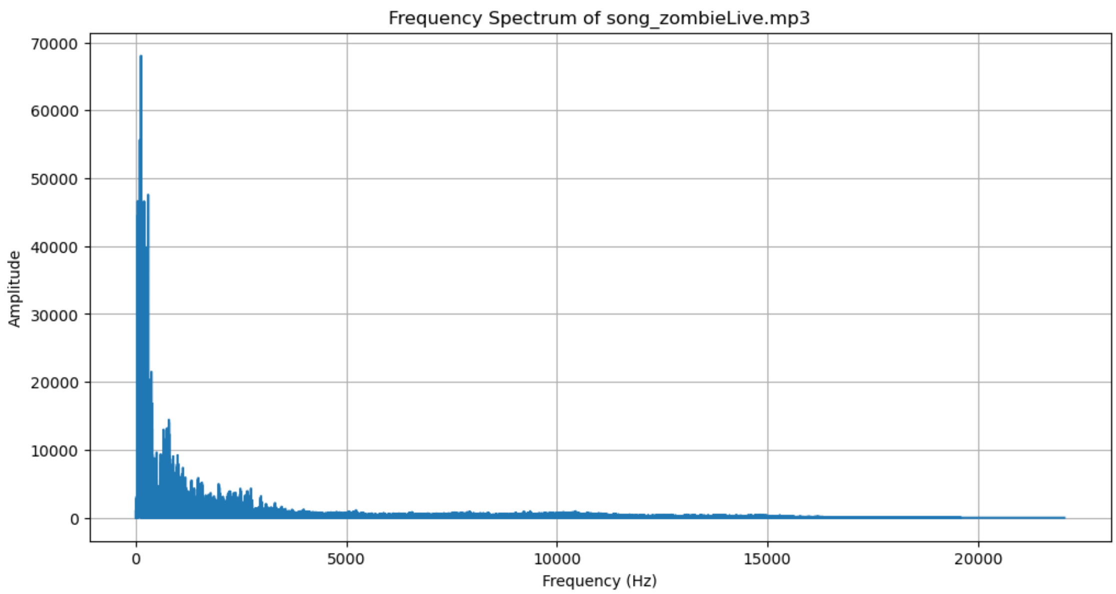
- Start working on frequency analysis of studio recorded music clips.
- Start working on time analysis for both live music and studio recorded music.
- Decide a way to use polynomial function to denoise live music while maintaining natural form

0.2.1 From Before(for code integrity):

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import glob
import os
import librosa

song_files = glob.glob('song_*.mp3')
for song_file in song_files:

    data, sample_rate = librosa.load(song_file, sr=None, mono=True)
    N = len(data)

    yf = np.fft.fft(data)
    xf = np.fft.fftfreq(N, 1 / sample_rate)
    idxs = np.where(xf >= 0)
    xf = xf[idxs]
    yf = np.abs(yf[idxs])

    plt.figure(figsize=(12, 6))
    plt.plot(xf, yf)
    plt.title(f'Frequency Spectrum of {os.path.basename(song_file)}')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
    plt.grid(True)
    plt.show()
```


Frequency Spectrum of song_radioHeadLive.mp3
Amplitude
20000
15000
10000
5000
0
0
5000
10000
15000
20000
Frequency (Hz)


Frequency Spectrum of song_coldplayLive.mp3
Amplitude
70000
60000
50000
40000
30000
20000
10000
0
0
5000
10000
15000
20000
Frequency (Hz)



0.3 New Progress

0.3.1 Import libraries and create studio to live recording pairs

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from scipy.signal import correlate
import os

import warnings
warnings.filterwarnings('ignore')
songs = [
    {'live': 'song_zombie.mp3', 'studio': 'zombie.wav'},
    {'live': 'song_creep.mp3', 'studio': 'creep.wav'},
    {'live': 'song_bittersweet.mp3', 'studio': 'bittersweet.wav'}
]
```

0.3.2 Process each song by:

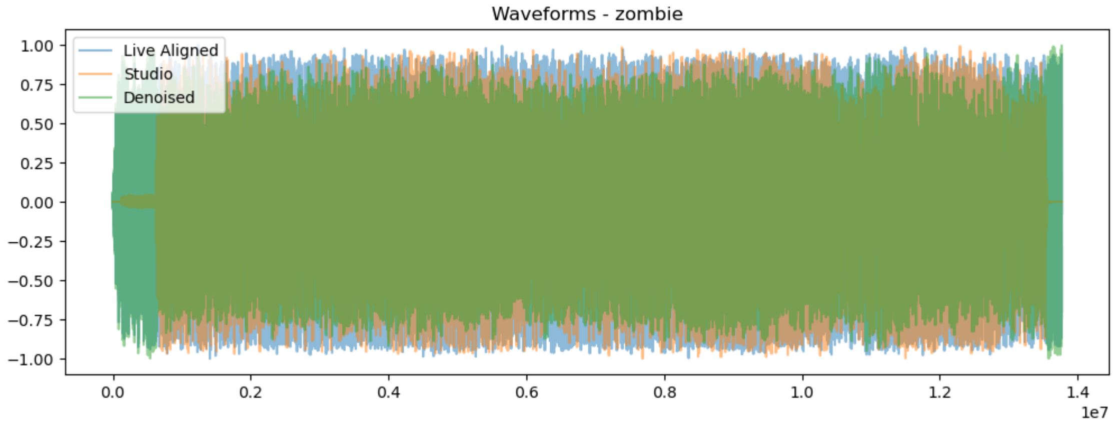
- Normalize the audio signals
- Time alignment using cross-correlation
- Align the live recording
- Truncate to the same length
- Compute STFT
- Compute magnitude and phase
- Spectral subtraction

- Reconstruct the denoised signal
- Normalize denoised audio

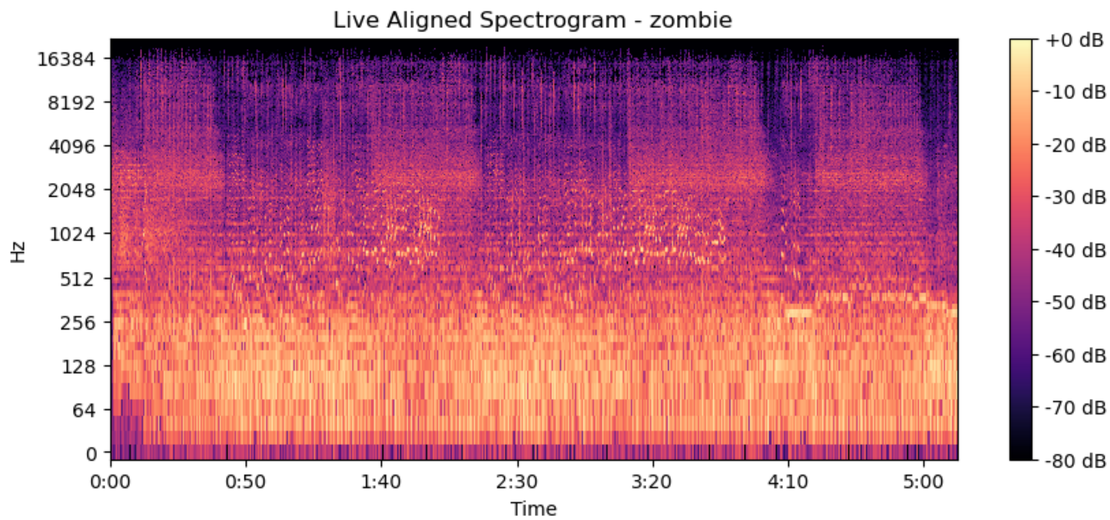
[ ]:
```python
for song in songs:
    live_file = song['live']
    studio_file = song['studio']
    song_name = os.path.splitext(os.path.basename(studio_file))[0]

    print(f"\nProcessing '{song_name}'...")
    data_live, sr_live = librosa.load(live_file, sr=None, mono=True)
    data_studio, sr_studio = librosa.load(studio_file, sr=None, mono=True)
    if sr_live != sr_studio:
        data_live = librosa.resample(data_live, sr_live, sr_studio)
        sr_live = sr_studio
    data_live = data_live / np.max(np.abs(data_live))
    data_studio = data_studio / np.max(np.abs(data_studio))
    print("  Performing time alignment...")
    correlation = correlate(data_live, data_studio, mode='full')
    lag = np.argmax(correlation) - len(data_studio) + 1
    if lag > 0:
        data_live_aligned = data_live[lag:]
    else:
        data_live_aligned = np.pad(data_live, (abs(lag), 0), 'constant')


    min_length = min(len(data_live_aligned), len(data_studio))
    data_live_aligned = data_live_aligned[:min_length]
    data_studio = data_studio[:min_length]


    print("  Computing STFT...")
    D_live = librosa.stft(data_live_aligned, n_fft=2048, hop_length=512)
    D_studio = librosa.stft(data_studio, n_fft=2048, hop_length=512)

    mag_live, phase_live = librosa.magphase(D_live)
    mag_studio, _ = librosa.magphase(D_studio)

    print("  Applying spectral subtraction...")
    mag_diff = mag_live - mag_studio
    mag_diff = np.maximum(mag_diff, 0)


    D_denoised = mag_diff * phase_live
    data_live_denoised = librosa.istft(D_denoised, hop_length=512)
    data_live_denoised = data_live_denoised / np.max(np.abs(data_live_denoised))
    output_file = f"denoised_{song_name}.wav"
    sf.write(output_file, data_live_denoised, sr_studio)
```

```python
    print(f"  Denoised audio saved to '{output_file}'.")

    # Plotting the results
    print("  Plotting the results...")
    plt.figure(figsize=(12, 4))
    plt.title(f"Waveforms - {song_name}")
    plt.plot(data_live_aligned, label='Live Aligned', alpha=0.5)
    plt.plot(data_studio, label='Studio', alpha=0.5)
    plt.plot(data_live_denoised, label='Denoised', alpha=0.5)
    plt.legend()
    plt.show()

    # Plot spectrograms
    def plot_spectrogram(data, sr, title):
        D = librosa.amplitude_to_db(np.abs(librosa.stft(data, n_fft=2048,␣
↪hop_length=512)), ref=np.max)
        plt.figure(figsize=(10, 4))
        librosa.display.specshow(D, sr=sr, hop_length=512, x_axis='time',␣
↪y_axis='log')
        plt.colorbar(format='%+2.0f dB')
        plt.title(title)
        plt.show()

    plot_spectrogram(data_live_aligned, sr_studio, f'Live Aligned Spectrogram -␣
↪{song_name}')
    plot_spectrogram(data_studio, sr_studio, f'Studio Spectrogram -␣
↪{song_name}')
    plot_spectrogram(data_live_denoised, sr_studio, f'Denoised Spectrogram -␣
↪{song_name}')

    print(f"Finished processing '{song_name}'.\n")
```

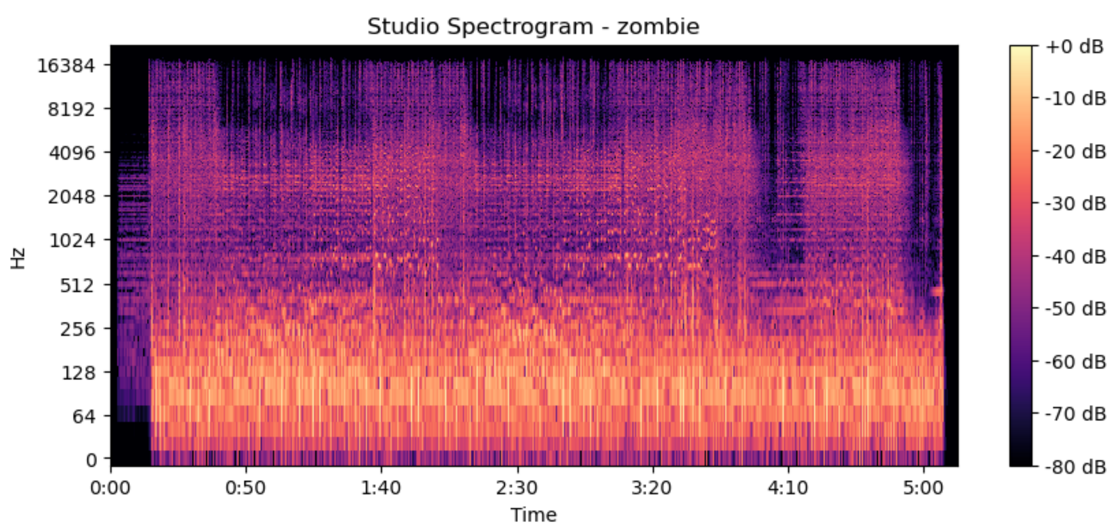
```
Processing 'zombie'…
  Performing time alignment…
  Computing STFT…
  Applying spectral subtraction…
  Denoised audio saved to 'denoised_zombie.wav'.
  Plotting the results…
```



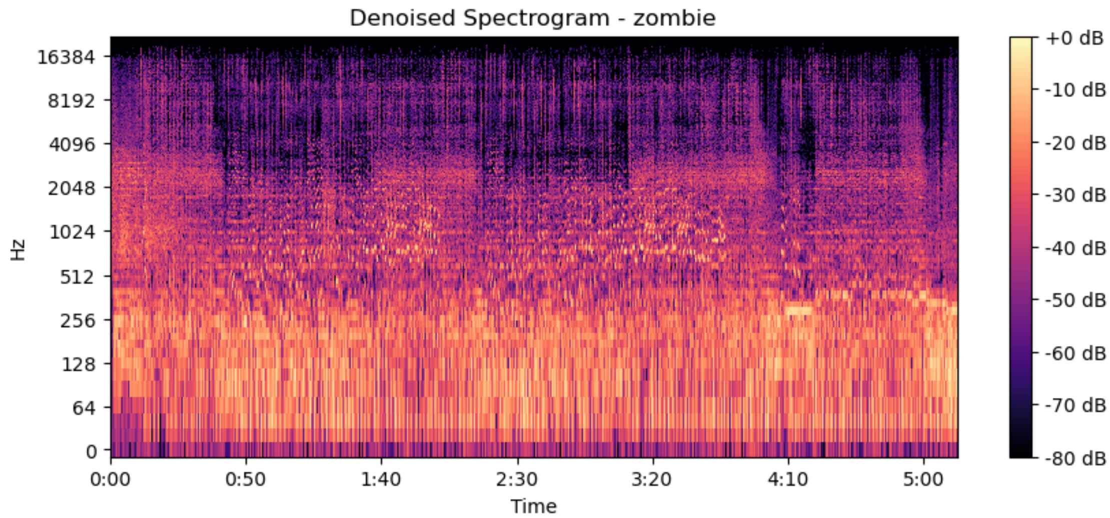
Waveforms - zombie
Live Aligned
Studio
Denoised
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e7



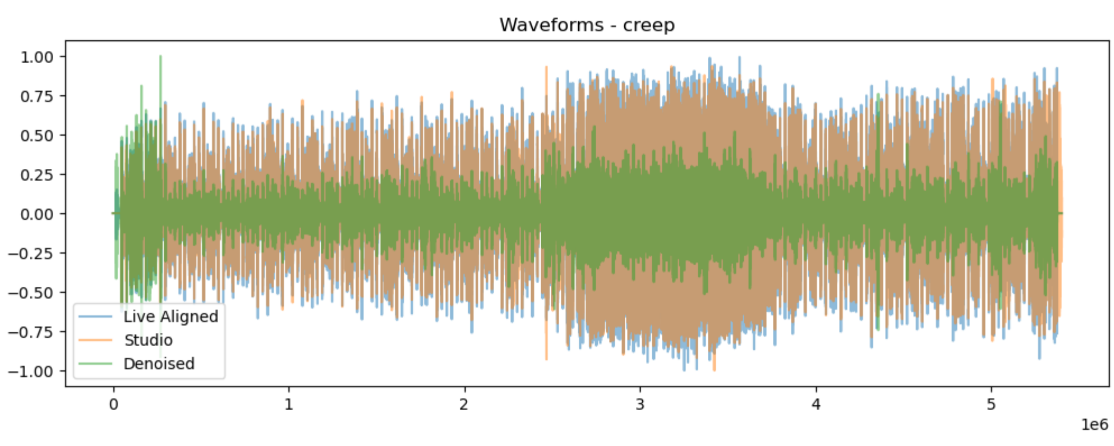
Live Aligned Spectrogram - zombie
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0:00
0:50
1:40
2:30
3:20
4:10
5:00
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

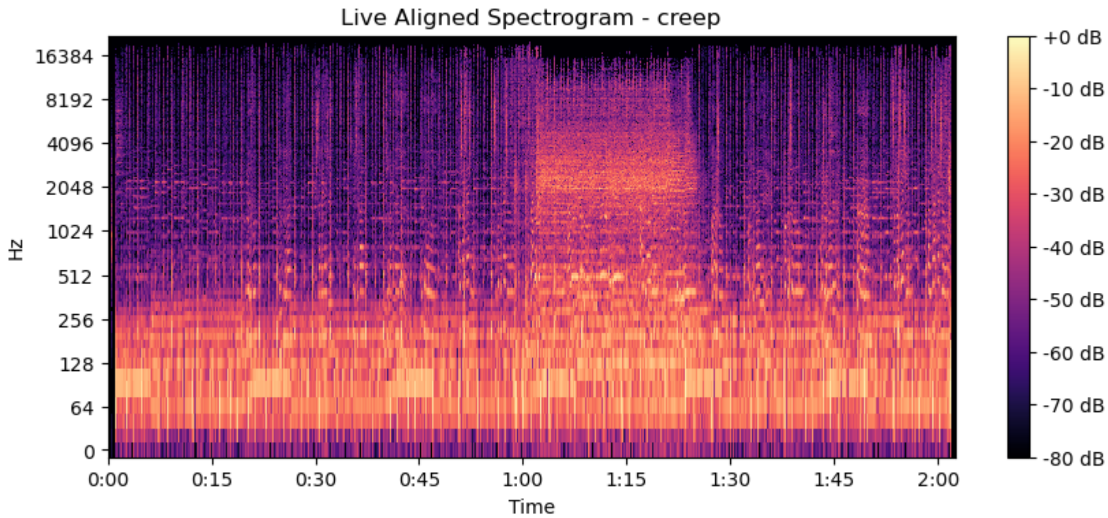


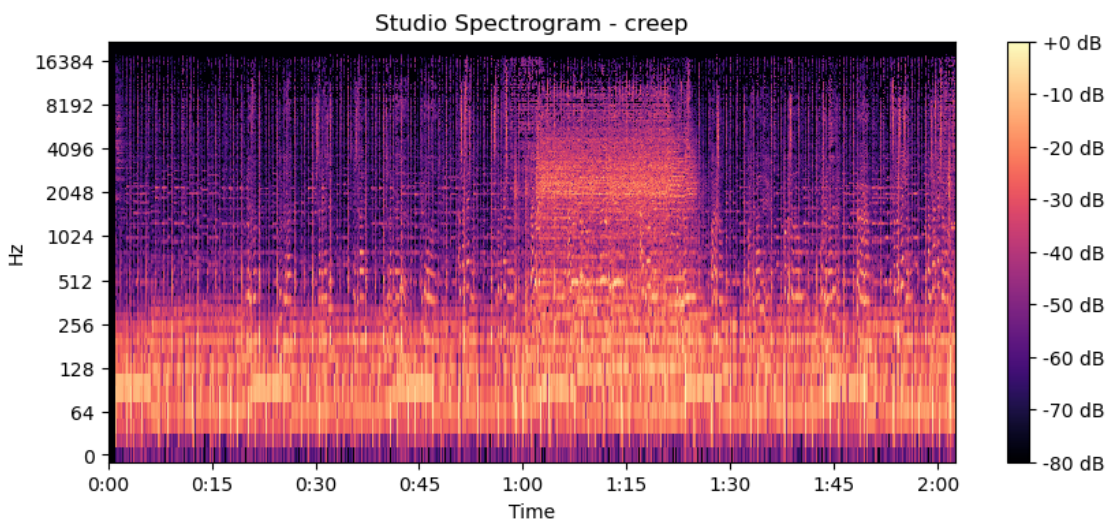
Studio Spectrogram - zombie
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0:00
0:50
1:40
2:30
3:20
4:10
5:00
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

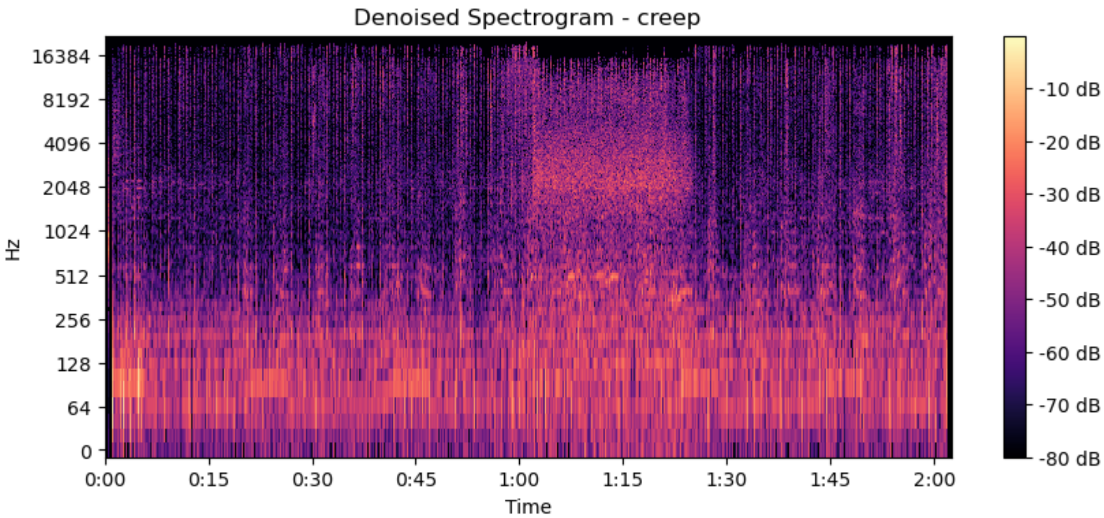



```
Finished processing 'zombie'.


Processing 'creep'…
  Performing time alignment…
  Computing STFT…
  Applying spectral subtraction…
  Denoised audio saved to 'denoised_creep.wav'.
  Plotting the results…
```

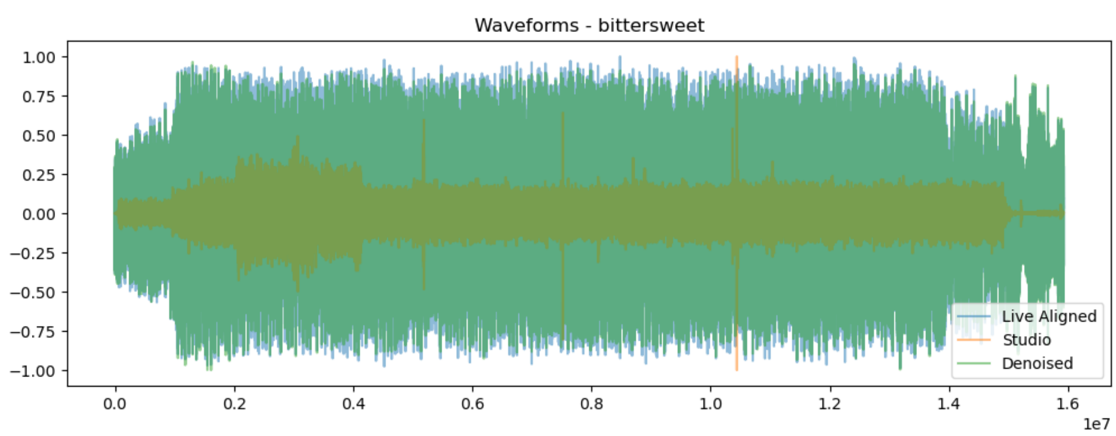


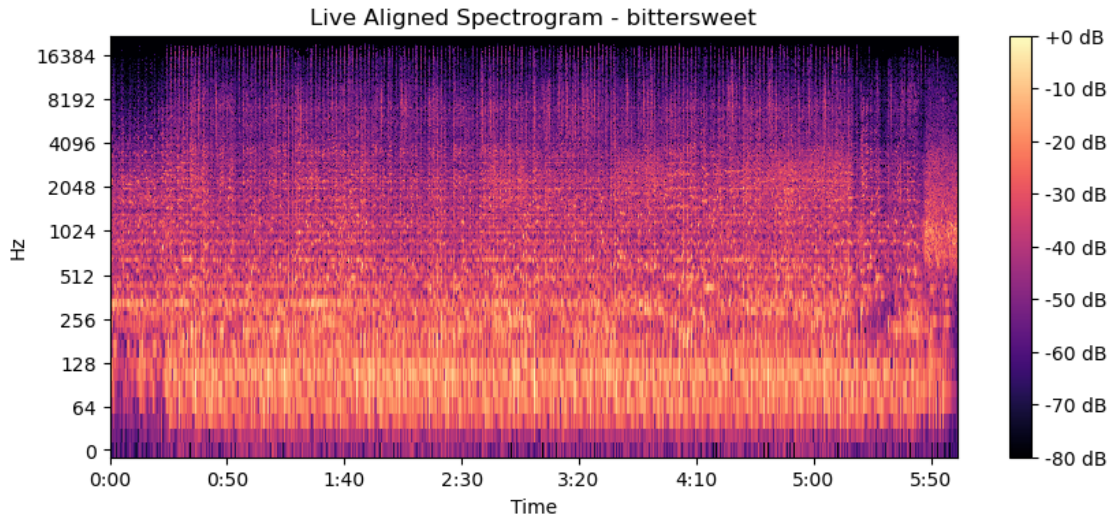

Live Aligned Spectrogram - creep
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0:00
0:15
0:30
0:45
1:00
1:15
1:30
1:45
2:00
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

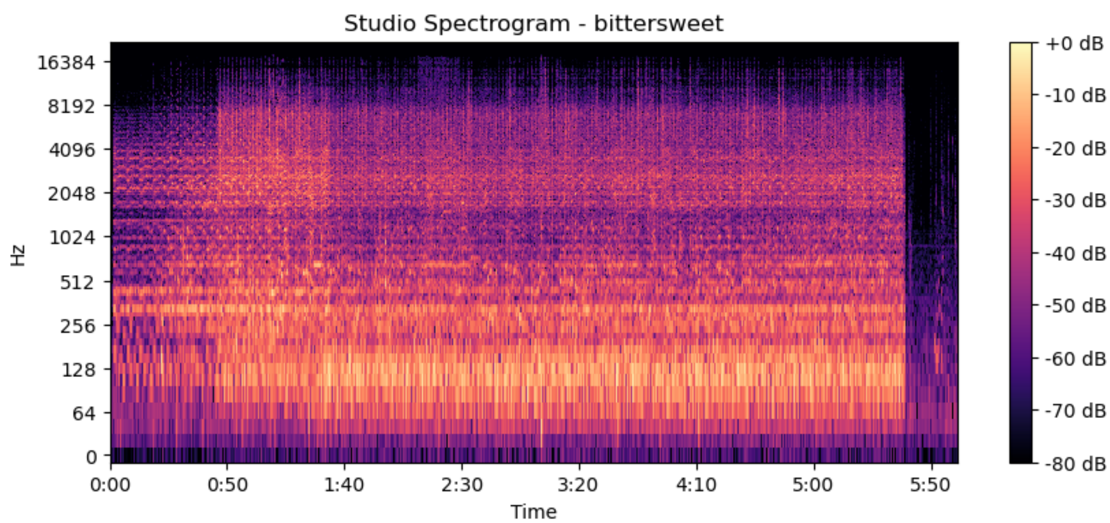

Studio Spectrogram - creep
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0:00
0:15
0:30
0:45
1:00
1:15
1:30
1:45
2:00
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

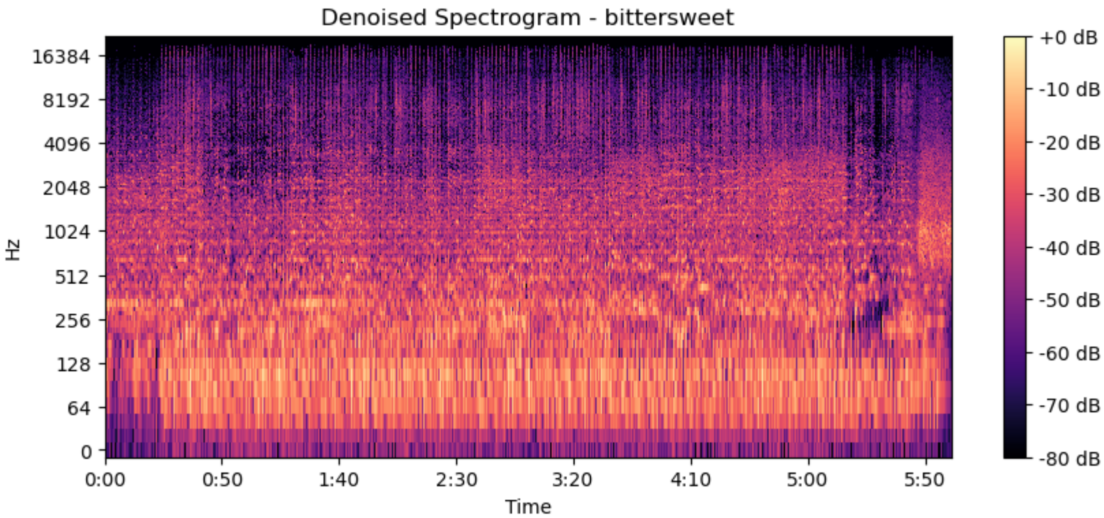



```
Finished processing 'creep'.


Processing 'bittersweet'...
  Performing time alignment...
  Computing STFT...
  Applying spectral subtraction...
  Denoised audio saved to 'denoised_bittersweet.wav'.
  Plotting the results...
```




Live Aligned Spectrogram - bittersweet
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0:00
0:50
1:40
2:30
3:20
4:10
5:00
5:50
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


Studio Spectrogram - bittersweet
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0:00
0:50
1:40
2:30
3:20
4:10
5:00
5:50
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB




```
Finished processing 'bittersweet'.
```

0.3.3 In the above charts it shows we achieved successful time and frequency alignment with live and studio recordings, however the normalization functions made denoise audio seem a little unnatural which would be improved in new updates.

0.4 Next Steps

- find alternative to normalizing functions
- find polynomial function for each song to denoise the live music
- find better way to improve the alignment between studio and live music