DSP Visualization Tool in Python

Introduction

DSP visualization is a popular topic, and there are many visualization tools in many languages, such as Python, MATLAB, and Javascript. In this mini project, I choose Python to implement visualization tool in my program. In my program, I use four Python libraries *NumPy*, *SciPy*, *wave*, *LibROSA*, and *Matplotlib*.

Visualization tools

- *Matplotlib* [1]

  Matplotlib is a very common and popular plotting library for the Python programming language and its numerical mathematics extension *NumPy*. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "*pylab*" interface based on a state machine, designed to closely resemble that of MATLAB, though its use is discouraged. *SciPy* makes use of *Matplotlib*.

- *LibROSA* [4]

  LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. And *LibROSA* has many useful functions, which can load audio input, compute mel spectrogram, MFCC, delta features, chroma, locate beat events, compute beat-synchronous features, and display features.

- *Pandas* [2]

  The *Pandas* name is derived from panel data, a term for multidimensional structured dataset. The *Pandas* library became highly optimized for usage, with useful written code that provides high function and rich data structured design. It is fast and easy to implement and contains a software library that is used within Python for powerful data analysis and manipulating data visualization.

- *Seaborn* [3]

  Seaborn is another open-source software library for data analysis and visualization. It is a popular library for making appealing statistical data graphs in Python. It provides high level of graphic interface for drawing platform and helps to easily identify patterns and draw attention to key elements. It includes built in themes for styling of informative data plots. It uses grid object method that connects the structure of the figure to the visual structure of the given data set. The main goal of this method is to simplify complicated plots.

Implementation

I define a function called “visualization”. I use this function to process sound (.wav file) signals. This function can compute the total number of frames, sampling frequency, the time interval between sampling points, the time of sound signal, and the amplitude of every frame of the total sound signal. The function also uses the FFT algorithm and

LibROSA for computing to get frequency domain information. The last step is use Matplotlib to visualize the sound signals in both time domain and frequency domain.

```
def visualization(filename):
    WAVE = wave.open(filename)
    x, sr = librosa.load(filename)
    X = librosa.stft(x)
    Xdb = librosa.amplitude_to_db(abs(X))
    for item in enumerate(WAVE.getparams()):
        print(item)
    a = WAVE.getparams().nframes
    f = WAVE.getparams().framerate
    sample_time = 1 / f
    time = a / f
    sample_frequency, audio_sequence = wavfile.read(filename)
    print(audio_sequence)
    x_seq = np.arange(0, time, sample_time)
    y_seq = abs(fft(audio_sequence))

    plt.plot(x_seq, audio_sequence, 'blue')
    plt.xlabel("time (s)")
    plt.ylabel("amplitude")
    plt.title("Time domain signal")

    plt.figure(figsize=(14, 5))
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()
    plt.show()

    plt.show()
```

Figure 1: The example code of function “visualization”

Experiment results

This program uses two sound files (“Canon.wav” and “Ring.wav”) to test.

1) “Canon.wav”

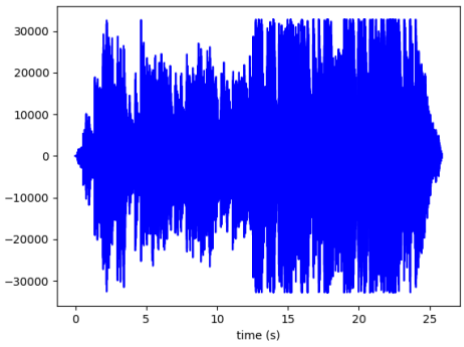

Figure 2: Sound signal of “Canon.wav”

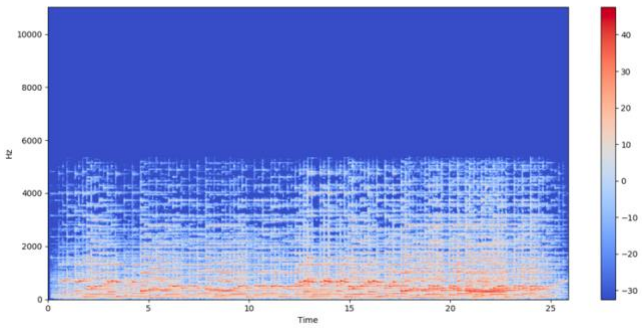

Figure 3: Spectrogram of “Canon.wav”

2) “Ring.wav”

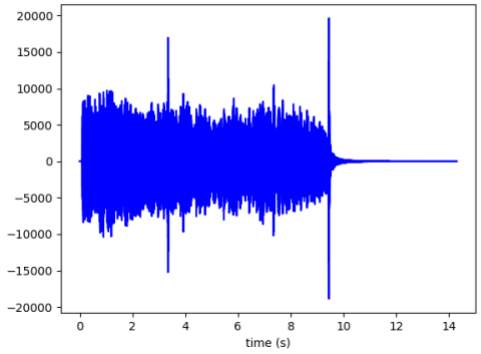

Figure 4: Sound signal of “Ring.wav”

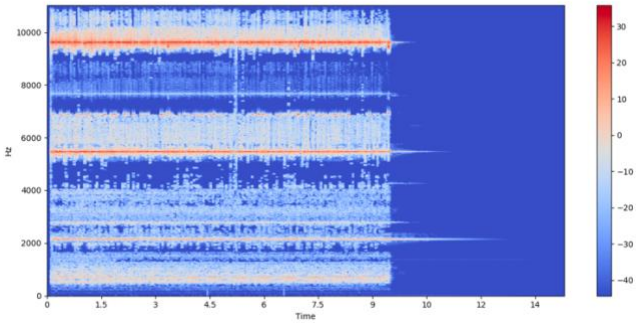

Figure 5: Spectrogram of “Ring.wav”

Reference

[1] "Matplotlib coding styles". matplotlib.org.

[2] https://pandas.pydata.org/

[3] https://seaborn.pydata.org/

[4] http://librosa.github.io/librosa/